COMPOSE: MOLECULAR GENERATION AND OPTIMIZATION WITH A REUSABLE STOCHASTIC REWRITE PROCESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Controllable molecular design requires models that describe how molecules can change and steer those changes toward desired outcomes. Existing models often specialize in particular endpoint distributions or task objectives, limiting their reuse as design goals change. To address this, we introduce **COMPOSE** (**CO**ntrollable **M**olecular **P**rocess **O**ver **S**tochastic **E**dits), which generates molecules through stochastic sequences of executable transformations between complete, chemically valid structures. We learn a goal-independent reference process over legal, variable-size transitions defined by an exact rewrite system. Structured programs combine these transitions into coordinated molecular transformations, while a separate controller steers them using task constraints and feedback. Keeping this reference process fixed, COMPOSE maintains 100% validity and achieved the highest mean quality among the compared methods across fragment-constrained generation tasks. This reuse extended to black-box optimization, where COMPOSE achieved a mean final top-10 score of 0.563 across 22 objectives, exceeding the strongest compared baseline's 0.542 under a 1,000-evaluation budget. In protein-specific lead optimization, COMPOSE obtained the best reported docking scores in 23 of 30 cases using 250 evaluations per lead, compared with 1,000 for the baselines. Together with similarity-constrained editing results, these findings show that reusable molecular transformations enable a single learned process to support generation and optimization as structural constraints and design objectives change, without retraining the underlying model.

1 INTRODUCTION

Generative modeling has become a central approach to molecular design, supporting both de novo generation and the optimization of existing molecules (Gómez-Bombarelli et al., 2018; Jin et al., 2018; Vignac et al., 2023; Lee et al., 2025). Recent graph-based diffusion, flow, and discrete generative models learn molecular distributions and support conditional generation toward desired structural and functional properties (Vignac et al., 2023; Campbell et al., 2024; Lee et al., 2025). The learned object is typically an endpoint distribution or a generator specialized toward a particular task. For molecular design, modeling the path between structures accounts for how each transformation changes both the next molecule and the set of molecular states that remain reachable afterward.

In molecular editing, each edit changes the set of transformations available from the resulting molecule. Existing editing and lead-optimization methods exploit sequential molecular modifications through graph-to-graph generation, reinforcement learning, fragment-based search, evolutionary optimization, and Markov chain Monte Carlo (MCMC) (Jin et al., 2019; Zhou et al., 2019; Jensen, 2019; Xie et al., 2021; Fu et al., 2021). An atom insertion, bond modification, or ring operation can open or eliminate downstream branches of the design space. These dependencies motivate modeling the transformations that connect molecular structures.

Learned generators can increasingly be guided toward desired properties, with generative trajectories often defined through corrupted, masked, or latent states (Vignac et al., 2023; Campbell et al., 2024; Lee et al., 2025). Molecular editing and search methods operate directly on complete molecules using proposal mechanisms typically prescribed by the optimizer (Jensen, 2019; Zhou et al., 2019;

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.

Xie et al., 2021). Recent work further shows that pretrained discrete generators can be redirected toward multiple objectives and constraints through inference-time guidance and control (Chen et al., 2025; 2026). Together, these advances motivate a goal-independent generative process over molecular transitions that can be learned once and controlled as design goals change.

Here, we introduce **COMPOSE** (**CO**ntrollable **M**olecular **P**rocess **O**ver **S**tochastic **E**dits), a generative method that learns a reusable molecular transition process over complete and chemically valid molecular graphs and controls it across design objectives (Figure 1). COMPOSE represents generation as a stochastic sequence of executable edits between complete molecules. At each molecular state, an exact executor enumerates the state-dependent set of legal, trans-dimensional molecular successors. We then learn a goal-independent reference law R_θ over this same successor set from executable molecular trajectories, capturing state-dependent preferences over molecular continuations without conditioning on a downstream design goal. Structured programs compose these edits into coordinated molecular transformations, and a separate controller adapts how those transformations are proposed and selected from task constraints and feedback. Because generation and control operate on the same explicit molecular support, the transition process can be learned once and redirected across design objectives without retraining the underlying generator.

We summarize our main contributions as follows.

- We introduce a stochastic molecular process over exact, executable, trans-dimensional rewrites between complete molecular graphs, with a learned reference law defined on canonical molecular successors.
- We introduce structured molecular programs that compose executable edits into coordinated transformations, together with a separate controller that adapts their use to task constraints and feedback while leaving the learned reference process fixed.
- We show that the reference process learns state-dependent transition preferences and transfers without retraining across fragment-constrained generation, similarity-constrained molecular editing, general black-box optimization, and protein-specific lead optimization.

2 COMPOSE

2.1 EXECUTABLE MOLECULAR STATE SPACE

COMPOSE represents a generative trajectory as a sequence of complete molecular graphs whose size may change over time. Let

$$\mathcal{X} = \bigcup_{n=1}^{n_{\max}} \mathcal{X}_n, \tag{1}$$

where $\mathcal{X}_n$ is the set of connected molecular graphs with n heavy atoms that satisfy the declared atom and bond vocabularies and valence constraints. Each state $x \in \mathcal{X}$ is therefore a complete molecule rather than a masked or partially specified representation.

Transitions between molecular states are defined by typed graph rewrites. An edit mark a specifies a rewrite together with its molecular operands and payload, and $T(x, a)$ denotes the resulting graph. From a state x, the executor admits

$$\mathcal{A}(x) = \{a : \ a \text{ matches } x,\ T(x, a) \in \mathcal{X}\}, \tag{2}$$

so every executable edit maps one molecular state in $\mathcal{X}$ to another. The rewrite system allows trajectories to grow, shrink, and restructure through atom-, bond-, and ring-level transformations. We give the full rewrite semantics and closure conditions in Sections C.1 and E.1, and describe coordinated multi-edit transformations in Section C.4.

2.2 LEARNING THE REFERENCE PROCESS

The executor determines which molecular transitions are possible, but not how probability should be distributed among them. We therefore learn a goal-independent reference law over executable edits. For an edit mark $a = (k, o, \eta)$, with rewrite family k, molecular operands o, and payload η, we factorize

$$p_\theta(a \mid x) = p_\theta(k \mid x)\, p_\theta(o, \eta \mid k, x), \qquad a \in \mathcal{A}(x). \tag{3}$$

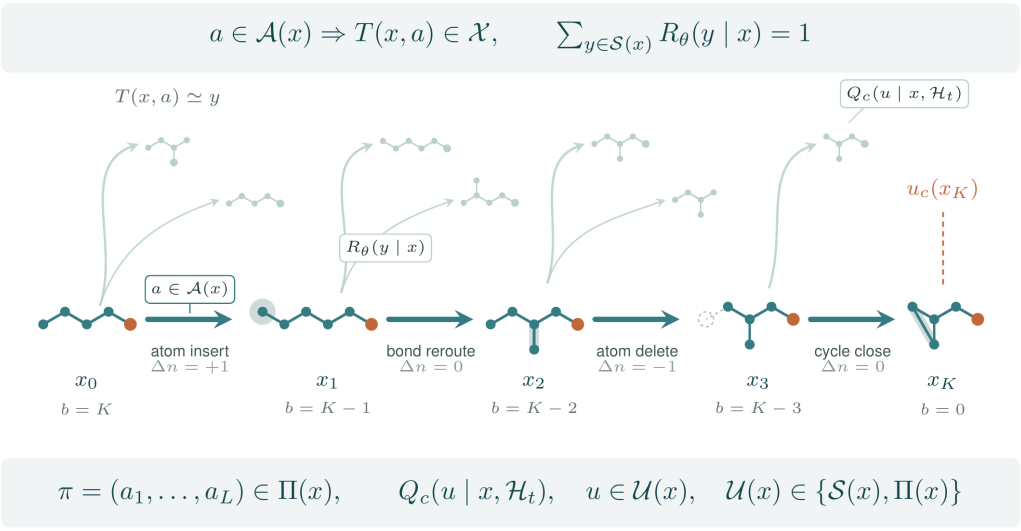


Figure 1: Overview of COMPOSE. The executor commits legal edits to complete molecular states, and R_θ is normalized over distinct canonical successors. The task-specific law Q_c selects a successor or executable program using task context c and feedback history $\mathcal{H}_t$ while the reference process remains fixed. The terminal score $u_c(x_K)$ illustrates an evaluated endpoint in feedback-driven optimization.

The distribution is normalized only over edits admitted by the executor and depends on the current molecular state, not on a downstream objective or task-specific constraint.

Different executable edits can produce the same molecule, so subsequent control acts on canonical molecular successors rather than edit marks. Let $\mathcal{S}(x)$ denote the distinct non-self successors of x, and let

$$\mathcal{G}_y(x) = \{a \in \mathcal{A}(x) : T(x,a) \simeq y\}$$

collect the edits that produce successor y. The induced reference kernel is

$$R_\theta(y \mid x) = \frac{\sum_{a \in \mathcal{G}_y(x)} p_\theta(a \mid x)}{\sum_{y' \in \mathcal{S}(x)} \sum_{a \in \mathcal{G}_{y'}(x)} p_\theta(a \mid x)}, \qquad y \in \mathcal{S}(x), \tag{4}$$

which removes dependence on symmetry or multiplicity in the underlying edit representation.

We learn R_θ from executable teacher transitions $x_i \to y_i^\star$ by minimizing

$$\mathcal{L}_{\text{ref}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log R_\theta(y_i^\star \mid x_i). \tag{5}$$

Every training transition is replayed through the same executor used at inference. Once trained, R_θ is fixed across all downstream design tasks.

2.3 Structured Molecular Programs

Individual rewrites define local molecular transitions, but many useful transformations require several dependent edits before a meaningful endpoint is reached. We represent such transformations as executable programs. A program

$$\pi = (a_1, \ldots, a_L)$$

induces the sequence

$$x^{(0)} = x, \qquad x^{(\ell)} = T\left(x^{(\ell-1)}, a_\ell\right), \qquad \ell = 1, \ldots, L,$$

and is executable from x when

$$a_\ell \in \mathcal{A}\left(x^{(\ell-1)}\right) \qquad \text{for all } \ell.$$

We write $T_\pi(x) = x^{(L)}$ for the resulting endpoint and $\Pi(x)$ for the set of programs executable from x.

Programs therefore compose transitions already admitted by the executor rather than introducing a new molecular state space or transition support. They allow dependent changes such as edits to bonds or rings created earlier in the same transformation to be proposed and executed as a single coordinated action. The primitive executor and learned reference process remain unchanged.

2.4 Control of the frozen molecular process

The reference process assigns probability to molecular transformations independently of any downstream design objective. We introduce task-specific control by selecting among executable programs. Let c denote the information supplied by a design problem and let $\mathcal{H}_t$ denote information accumulated after t control decisions. A controller defines

$$C(\pi \mid x, c, \mathcal{H}_t), \qquad \pi \in \Pi(x), \tag{6}$$

a distribution over programs executable from the current molecular state x.

A selected program produces the next state through

$$y = T_\pi(x).$$

The task context c specifies what the controller should preserve or optimize, while $\mathcal{H}_t$ may contain information revealed during search, such as previously evaluated molecules and their objective values. The executor and reference process remain fixed, so different design problems are expressed by changing the controller rather than relearning the underlying molecular process.

We next instantiate this formulation for structural conditioning and feedback-driven molecular optimization.

2.5 Structural conditioning

Some design problems specify molecular structure that must be preserved while the remainder of the molecule is generated or modified. We represent this information by

$$c_{\mathrm{str}} = (\mathcal{L}, \partial\mathcal{L}, \Gamma),$$

where $\mathcal{L}$ is the retained subgraph, $\partial\mathcal{L}$ identifies the interfaces through which it may be extended, and Γ specifies any required connectivity between those interfaces. This restricts the executable program set to

$$\Pi_{c_{\mathrm{str}}}(x) = \{\pi \in \Pi(x) : \pi \text{ preserves } \mathcal{L} \text{ and satisfies } \Gamma\}. \tag{7}$$

The structural controller places probability only on programs in $\Pi_{c_{\mathrm{str}}}(x)$ and conditions program construction on the retained structure and its allowed interfaces. Different fragment-constrained tasks therefore correspond to different choices of c_{str}, while the molecular state space, executor, and learned reference process remain unchanged.

2.6 Feedback-driven molecular optimization

For black-box objectives, the controller adapts using molecular endpoints evaluated during search. After t objective evaluations, we maintain the archive

$$\mathcal{D}_t = \{(x_i, \pi_i, y_i, u_c(y_i))\}_{i=1}^{t}, \qquad y_i = T_{\pi_i}(x_i), \tag{8}$$

where $u_c(y)$ is the task objective, oriented so that larger values are better. The controller uses $\mathcal{D}_t$ to select previously evaluated molecules as parents and construct new executable programs, which are evaluated only after execution and any inexpensive endpoint constraints have been checked. The resulting observations are then added to the archive.

For similarity-constrained lead optimization, the task additionally specifies a starting molecule x_0 and a feasible endpoint region

$$\mathcal{F}(x_0, \delta) = \{x \in \mathcal{X} : \mathrm{sim}(x, x_0) \geq \delta,\ \mathrm{QED}(x) \geq 0.6,\ \mathrm{SA}(x) \leq 4\}. \tag{9}$$

Only endpoints in $\mathcal{F}(x_0, \delta)$ are passed to the docking objective. The same executor and frozen reference process therefore support both unconstrained black-box optimization and optimization within an explicitly defined molecular region.

2.7 FINITE-HORIZON FUTURE-VALUE CONTROL

Some design problems require choosing an edit not only by the molecule it produces immediately, but by the molecular states that remain reachable afterward. For a design goal z, let $g_z(x)$ denote the desirability of a terminal state. With b transitions remaining, define

$$h_0(x, z) = g_z(x), \qquad h_b(x, z) = \sum_{y \in \mathcal{S}(x)} R_\theta(y \mid x)\, h_{b-1}(y, z). \tag{10}$$

Thus, $h_b(x, z)$ measures the expected terminal desirability reachable from x under the frozen reference process.

These future values induce the controlled transition

$$P^{z,b}(y \mid x) = R_\theta(y \mid x) \frac{h_{b-1}(y, z)}{h_b(x, z)}, \tag{11}$$

which favors successors with more desirable continuations while remaining on the support of R_θ.

Exact future values are intractable at molecular scale, so we learn $h_\phi(x, z, b) \approx h_b(x, z)$ from finite-horizon trajectories of the frozen reference process and substitute it into Equation (11). We use the resulting future-guided kernel within sequential Monte Carlo for similarity-constrained QED editing, while keeping R_θ fixed across sources and objectives. Details of the approximation, sampling procedure, and exact h-transform identity are given in Section D.2.

3 EXPERIMENTS

We evaluated whether COMPOSE learns a molecular process that can be reused across different design problems without retraining. We first tested whether the learned reference process captures state-dependent preferences among executable molecular successors. We then kept the GuacaMol-trained reference process and rewrite system fixed across fragment-constrained generation, similarity-constrained molecular editing, general black-box optimization, and protein-specific lead optimization. In each setting, only the task information and the controller used to steer the underlying process changed.

3.1 COMPOSE LEARNS STATE-DEPENDENT EDIT PREFERENCES

We tested whether R_θ learned state-dependent preferences among executor-admitted molecular successors. On 3,545 held-out transitions whose source molecules were disjoint from training, we compared R_θ with a frequency baseline that had access to the same executable edits and canonical successors but assigned probability using only edit-family frequencies from the training data.

On this held-out set, R_θ reduced canonical-successor negative log-likelihood (NLL) from 5.37 to 3.90 nats. With edit-family probabilities fixed to their training frequencies, learned state-dependent successor preferences recovered 86.7% of this reduction (Table 6). We used the same learned R_θ, without further training, in all subsequent experiments. Results restricted to edit families with data-backed supervision are reported in Section G.1.

3.2 COMPOSE PERFORMS COMPETITIVELY ON FRAGMENT-CONSTRAINED GENERATION

To evaluate structural conditioning, we generated complete molecules for five fragment tasks with retained substructures and task-specific attachment or connectivity requirements. The benchmark introduced with SAFE-GPT (Noutahi et al., 2024) and subsequently adopted by GenMol and InVirtuo-Gen spans motif extension, scaffold decoration, linker design, scaffold morphing, and superstructure generation, ranging from growth around a retained fragment to generation between fixed molecular boundaries.

COMPOSE handles these constraints directly through its transition process. Required atoms and bonds are protected, and the controller selects only executable edits or programs consistent with the specified attachment or connectivity conditions. We used the same GuacaMol-trained R_θ and rewrite system across all five tasks, with task-specific structural conditioning and program construction. We report quality, uniqueness, diversity, and validity following the benchmark protocol.

Across the five fragment-constrained tasks, COMPOSE achieved the highest reported quality on superstructure generation, linker design, and scaffold morphing while maintaining 100% validity across all settings (Table 1). Motif-extension quality exceeded both published baselines. For scaffold decoration, COMPOSE achieved the highest mean quality and uniqueness, while GenMol had higher diversity.

To test whether the learned probability distribution of the reference process improves generation beyond the executable rewrite system itself, we replaced its learned probabilities over available edit families and actions with uniform choices in the superstructure task, while keeping the executor, structural constraints, and stopping dynamics fixed. Both variants maintained 100% chemical validity and prompt fidelity, but removing the learned transition preferences reduced mean quality from 39.03% to 25.40% and uniqueness from 97.33% to 90.17%. These results show that performance is not explained by the validity-preserving transition space alone and that the learned reference process provides useful preferences over which executable molecular transitions to take (Section G.3).

We next asked whether the reference also improves selection after executable candidate programs have already been constructed. Using identical locked proposal panels, with proposal construction, structural checks, and candidate endpoints held fixed, reference-based selection increased mean quality from 37.20% to 42.63% in motif extension and from 34.47% to 36.70% in scaffold decoration. In linker design, combining reference scores with a preference for previously unseen outputs increased quality from 24.00% to 31.53%. These comparisons show that selection contributes beyond the availability of executable candidate completions (Section H.5).

Task	Metric	**COMPOSE**	InVirtuoGen	GenMol
Motif extension	Quality ↑	**42.63±0.38**	39.27±1.078	30.1±0.4
	Uniqueness ↑	93.63±0.60	**96.83±0.290**	77.5±0.1
	Diversity ↑	**0.664±0.001**	0.620±0.005	0.617±0.002
	Validity ↑	**100.00±0.00**	68.97±0.759	82.9±0.1
Superstructure	Quality ↑	**39.03±3.09**	27.43±0.953	34.8±1.0
	Uniqueness ↑	97.33±0.61	**99.41±0.157**	83.6±1.0
	Diversity ↑	0.725±0.001	**0.730±0.001**	0.599±0.009
	Validity ↑	**100.00±0.00**	75.70±0.898	97.5±0.9
Linker design	Quality ↑	**31.53±1.40**	22.33±1.250	4.3±0.4
	Uniqueness ↑	81.03±0.35	84.76±1.620	**93.9**
	Diversity ↑	**0.564±0.002**	0.520±0.004	0.529±0.002
	Validity ↑	**100.00±0.00**	60.37±0.573	16.7±0.2
Scaffold morphing	Quality ↑	**31.53±1.40**	22.33±1.250	4.3±0.4
	Uniqueness ↑	81.03±0.35	84.76±1.620	**93.9**
	Diversity ↑	**0.564±0.002**	0.520±0.004	0.529±0.002
	Validity ↑	**100.00±0.00**	60.37±0.573	16.7±0.2
Scaffold decoration	Quality ↑	**36.70±1.80**	**36.37±1.096**	31.8±0.5
	Uniqueness ↑	**93.67±0.83**	88.58±1.130	82.7±1.8
	Diversity ↑	0.563±0.002	0.560±0.003	**0.591±0.001**
	Validity ↑	**100.00±0.00**	90.70±0.616	96.6±0.8

Table 1: **Fragment-constrained generation.** Quality, uniqueness, and validity are reported as percentages, and diversity as a fraction (higher is better for all metrics). Values are means ± standard deviations over three runs where reported. The shaded column shows COMPOSE results, with published InVirtuoGen and original GenMol results shown for comparison. Bold indicates the highest mean and any result whose mean ± one-standard-deviation interval overlaps that of the highest mean. Bolding does not imply statistical significance. Scaffold-morphing results reuse the respective linker outputs and do not constitute independent experiments. We use GenMol's one-step linker results, following InVirtuoGen's comparison. GenMol's repository reports its one-step linker uniqueness as 93.9% without a standard deviation.

3.3 COMPOSE SUPPORTS SIMILARITY-CONSTRAINED MOLECULAR EDITING

We tested whether learned future values could guide COMPOSE toward higher-QED molecules that retain source similarity. The ZINC-250k editing benchmark used by GrIDDD (Ninniri et al., 2025) starts from molecules with QED in $[0.7, 0.8]$ and requires QED ≥ 0.9 while maintaining Morgan-fingerprint Tanimoto similarity ≥ 0.4 to the source. We left the GuacaMol-trained R_θ unchanged and applied the finite-horizon controller from Section 2.7. The learned future value h_ϕ favors states predicted to reach the target region within the remaining horizon, and twisted SMC concentrates computation on those continuations while every transition remains an executable COMPOSE edit. Each of the K returned molecules is the terminal state of one controlled trajectory.

On our 800-source evaluation panel, COMPOSE reached the target region for 446 sources (55.8%, 95% confidence interval $[52.3, 59.2]$) with $K = 8$ returned trajectories per source. GrIDDD reported 45.1% success with $K = 20$ on a different source panel, so this comparison is unpaired. The tabulated values and evaluator conventions are in Table 10 and Section I.1.

To isolate the role of future-aware control, we also compared local and future-aware selection on 65 held-out source-target pairs with a known executable solution path. Under the same edit budget and successor set, local selection reached 26 targets, whereas future-aware control reached 40. Using h_ϕ to prioritize a single successor preserved the 40/65 recovery rate while evaluating 34.6% of the continuations required by exhaustive look-ahead (Section H.1).

Table 2: **Practical Molecular Optimization under a 1,000-call budget (PMO-1K).** Final top-10 score after 1,000 oracle evaluations, with higher values indicating better performance. Baseline results are the base no-memory results reported by Yang et al. (2026). Values are mean ± standard deviation over three seeds. The mean follows their 22-objective protocol and excludes valsartan SMARTS. Bold marks the best result in each row.

Oracle	**COMPOSE**	GenMol	InVirtuoGen	REINVENT	Graph-GA
Albuterol similarity	**0.782±0.020**	0.567±0.01	0.566±0.01	0.583±0.03	0.508±0.06
Amlodipine MPO	0.539±0.035	**0.571±0.01**	0.536±0.01	0.496±0.01	0.539±0.01
Celecoxib rediscovery	0.335±0.053	**0.456±0.01**	0.413±0.03	0.426±0.03	0.385±0.01
Deco hop	0.586±0.010	**0.641±0.03**	0.612±0.00	0.591±0.01	0.607±0.01
DRD2	0.979±0.018	0.984±0.00	0.993±0.01	0.975±0.01	**0.999±0.01**
Fexofenadine MPO	**0.751±0.043**	0.731±0.01	0.702±0.02	0.695±0.01	0.699±0.00
GSK3B	0.535±0.065	0.673±0.01	0.572±0.09	**0.773±0.08**	0.590±0.03
Isomers C7H8N2O2	**0.911±0.029**	0.485±0.05	0.597±0.05	0.879±0.04	0.846±0.02
Isomers C9H10N2O2PF2Cl	**0.895±0.064**	0.553±0.04	0.672±0.10	0.696±0.05	0.723±0.04
JNK3	0.245±0.041	**0.492±0.01**	0.484±0.08	0.205±0.02	0.244±0.03
Median1	**0.294±0.029**	0.236±0.00	0.243±0.01	0.236±0.02	0.222±0.01
Median2	0.181±0.012	**0.244±0.00**	0.187±0.02	0.194±0.00	0.211±0.01
Mestranol similarity	**0.503±0.013**	0.438±0.01	0.408±0.02	0.354±0.04	0.399±0.03
Osimertinib MPO	**0.833±0.027**	0.815±0.00	0.783±0.00	0.776±0.01	0.793±0.00
Perindopril MPO	**0.525±0.018**	0.517±0.01	0.454±0.01	0.417±0.01	0.471±0.01
QED	**0.945±0.002**	0.931±0.01	0.936±0.00	0.940±0.00	0.939±0.00
Ranolazine MPO	**0.770±0.052**	0.686±0.01	0.687±0.02	0.639±0.01	0.646±0.02
Scaffold hop	0.461±0.007	**0.506±0.00**	0.486±0.01	0.467±0.00	0.473±0.01
Sitagliptin MPO	**0.343±0.009**	0.228±0.02	0.338±0.04	0.310±0.00	0.332±0.02
Thiothixene rediscovery	0.280±0.038	**0.397±0.01**	0.342±0.02	0.326±0.02	0.323±0.01
Troglitazone rediscovery	0.314±0.057	**0.317±0.01**	0.266±0.01	0.260±0.00	0.287±0.01
Zaleplon MPO	0.385±0.066	**0.460±0.01**	0.436±0.00	0.448±0.00	0.432±0.01
Mean	**0.563**	0.542	0.532	0.531	0.530

3.4 COMPOSE ADAPTS TO DIVERSE BLACK-BOX OBJECTIVES

We evaluated feedback-driven molecular optimization across diverse objectives using only scores collected during search, without a supplied target structure or starting lead. The Practical Molecular Optimization (PMO) benchmark (Gao et al., 2022) spans similarity and rediscovery, scaffold and substructure objectives, multi-property optimization, drug-likeness, and predicted biological activity. We followed the 1,000-call protocol of Yang et al. (2026) in a no-prescreen setting. Every objective began from the same task-independent initialization bank, and no target-oracle scores were used outside the evaluation budget. The GuacaMol-trained R_θ and rewrite system remained fixed, and the controller adapted only from scores observed during each run.

Across the 22 objectives in the reported protocol, COMPOSE achieved the highest mean final top-10 score among the compared methods and led on 11 individual objectives (Table 2). Task-level leads included albuterol similarity, both isomer objectives, mestranol similarity, and several multi-property objectives. The breadth of task-level gains supports the use of COMPOSE's shared executable program space for diverse objectives, although controller performance remained task dependent. Across the same objectives, COMPOSE's mean AUC-Top10 was 0.482, below GenMol's published base-generator value of 0.537 (Table 11).

To test whether structured molecular programs improve optimization beyond simply permitting multiple legal edits, we replaced the structured proposer with uniform legal edit chains under a matched primitive-edit allowance. Across 14 completed matched objective-seed pairs, mean final Top-10 decreased from 0.621 to 0.558 and mean AUC-Top10 from 0.503 to 0.465. The effect varied across objectives, but COMPOSE achieved the higher value in 10 of 14 pairs for each metric (Section H.6).

3.5 COMPOSE IMPROVES DOCKING SCORES UNDER LEAD-SIMILARITY CONSTRAINTS

We next evaluated COMPOSE on protein-specific lead optimization. We used the benchmark introduced by GenMol and later considered by InVirtuoGen, with five protein targets, three starting molecules per target, and similarity thresholds $\delta \in \{0.4, 0.6\}$. Feasible molecules must satisfy QED ≥ 0.6, SA ≤ 4, and the required Morgan-fingerprint Tanimoto similarity to the starting molecule. We kept the GuacaMol-trained R_θ and rewrite system fixed and used the feedback-driven controller from Section 2.6, evaluating docking scores only for feasible endpoints.

With 250 docking evaluations per starting molecule, COMPOSE obtained the best feasible docking score in 10 of 15 target-molecule cases at $\delta = 0.4$ and 13 of 15 at $\delta = 0.6$ (Table 3). InVirtuoGen and GenMol used 1,000 evaluations per starting molecule. The higher count at $\delta = 0.6$ indicates that COMPOSE remained effective as the admissible endpoint region narrowed. The PMO and T4 results together support feedback-driven use of the same frozen molecular process for optimization from a task-independent bank and from starting leads subject to endpoint constraints.

	Seed	$\delta = 0.4$			$\delta = 0.6$		
Target	score	**COMPOSE**	IVG	GenMol	**COMPOSE**	IVG	GenMol
PARP1	-7.3	-11.9	**-14.1**	-10.6	**-13.6**	-12.3	-10.4
	-7.8	-12.3	**-13.4**	-11.0	**-12.7**	-11.7	-9.7
	-8.2	**-14.1**	-9.0	-11.3	**-11.3**	-10.7	-9.2
FA7	-6.4	**-9.3**	-8.4	-8.4	-6.4	**-7.7**	-7.3
	-6.7	**-9.4**	-8.9	-8.4	**-7.8**	-7.5	-7.6
	-8.5	**-9.6**	-8.0	-	**-8.5**	-7.4	-
5HT1B	-4.5	-12.4	**-13.3**	-12.9	**-13.3**	-12.4	-12.1
	-7.6	-11.9	-12.0	**-12.3**	**-12.2**	-12.0	-12.0
	-9.8	**-12.5**	-10.9	-11.6	**-10.8**	-10.6	-10.5
BRAF	-9.3	**-12.2**	-10.1	-10.8	-9.0	**-9.7**	-
	-9.4	-10.5	**-10.8**	**-10.8**	**-11.0**	-10.4	-9.7
	-9.8	**-10.9**	-10.6	-10.6	**-11.5**	-10.3	-10.5
JAK2	-7.7	**-10.4**	-10.2	-10.2	**-10.6**	-9.7	-9.3
	-8.0	**-11.3**	-10.5	-10.0	**-10.8**	-10.4	-9.4
	-8.6	**-10.7**	-10.2	-9.8	**-10.8**	-10.3	-

Table 3: **Similarity-constrained lead optimization.** Best feasible QuickVina2 docking score for each seed and similarity threshold, with lower values indicating better predicted binding. Feasible molecules satisfy QED ≥ 0.6, SA ≤ 4, and Tanimoto similarity $\geq \delta$ to the starting molecule. COMPOSE uses 250 docking evaluations per seed, compared with 1,000 for IVG and GenMol. Bold marks the best result in each row. A hyphen denotes an unreported baseline result.

4 DISCUSSION

In this work, we introduce COMPOSE, a generative framework that frames molecular design as control over a reusable stochastic process of executable molecular transformations rather than centering generation on a task-specific endpoint distribution. Across structural conditioning and constrained molecular editing, the learned reference process is controlled without retraining. Black-box optimization and protein-directed lead optimization search coordinated programs through the same executable rewrite system with the reference checkpoint unchanged. Task-specific control over this fixed executable process enabled generation and optimization under different structural and property objectives. Ablations further showed that the learned reference process improves molecular generation and that structured molecular programs provide a clear advantage in optimization.

The current framework is limited by the molecular transformations represented by its executable state space and by the quality of the objectives used for downstream control. Chemical validity alone does not translate to synthetic accessibility or experimental activity, and complex structural changes may require coordinated transformations that are difficult to discover through local edits. Extending the executable process and improving control over such transformations could broaden the range of molecular-design problems accessible within the same framework. Ultimately, COMPOSE provides a foundation for molecular design in which a single learned process supports generation, editing, and lead optimization as structural constraints and design objectives change.

AI USE STATEMENT

Generative AI tools assisted with conceptual and methodological development, software implementation and debugging, experimental design and diagnostic analysis, interpretation of computational results, literature search and summarization, and drafting and editing the manuscript. The authors are responsible for the final code, reported results, citations, and conclusions, including material produced with AI assistance.

ETHICS STATEMENT

This work evaluates molecular generation and optimization computationally; the generated compounds are not experimentally validated. A molecular design system can be misused, and chemical validity, synthetic-accessibility scores, predicted properties, and docking scores do not establish safety or biological efficacy. We therefore interpret outputs as computational candidates rather than validated compounds for synthesis or use.

REPRODUCIBILITY STATEMENT

The Methods and Appendix specify the molecular representation, executable rewrite system, learned reference process, controllers, benchmark protocols, and evaluation metrics. Appendix J documents the software environment, randomization, compute, and provenance required to reproduce the reported analyses, including the access limitations of bulk artifacts. For reproducibility, we host our code, data-processing scripts, and evaluation pipelines at this anonymous repository: `https://anonymous.4open.science/r/compose_rgm-6336/`.

REFERENCES

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning (ICML)*, 2024.

Tong Chen, Yinuo Zhang, and Pranam Chatterjee. AReUReDi: Annealed rectified updates for refining discrete flows with multi-objective guidance. *arXiv preprint arXiv:2510.00352*, 2025.

Tong Chen, Maximilian Holsman, Lin Zhao, Yinuo Zhang, and Pranam Chatterjee. pCoMole: Pareto-constrained molecule editing with discrete flows. OpenReview, 2026. OpenReview submission `https://openreview.net/forum?id=LpttbUObQL`.

Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M. Glass, and Jimeng Sun. MIMOSA: Multi-constraint molecule sampling for molecule optimization. In *AAAI Conference on Artificial Intelligence*, 2021.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1551. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8644353f7d307baaf29bc1e56fe8e0ec-Abstract-Datasets_and_Benchmarks.html.

Shrey Goel and Pranam Chatterjee. Minimal-action discrete Schrödinger bridge matching for peptide sequence design. *arXiv preprint arXiv:2601.22408*, 2026.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

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

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

Cristian Jensen, Luca Schaufelberger, Riccardo De Santi, Kjell Jorner, and Andreas Krause. Value matching: Scalable and gradient-free reward-guided flow adaptation. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview https://openreview.net/forum?id=7iXt44Actj.

Jan H. Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10(12):3567–3572, 2019.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning (ICML)*, 2018.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1812.01070.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 4839–4848, 2020.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Gopal Paliwal, Weili Nie, and Arash Vahdat. GenMol: A drug discovery generalist with discrete diffusion. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 33205–33226, 2025. URL https://proceedings.mlr.press/v267/lee25o.html.

Matteo Ninniri, Marco Podda, and Davide Bacciu. Graph diffusion that can insert and delete. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.15725.

Yifan Niu, Ziqi Gao, Tingyang Xu, Yang Liu, Yatao Bian, Yu Rong, Junzhou Huang, and Jia Li. InversionGNN: A dual path network for multi-property molecular optimization. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2503.01488.

Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan S. C. Lim, and Prudencio Tossou. Gotta be SAFE: a new framework for molecular design. *Digital Discovery*, 3(4):796–804, 2024. doi: 10.1039/D4DD00019F.

Sophia Tang, Yinuo Zhang, and Pranam Chatterjee. PepTune: De novo generation of therapeutic peptides with multi-objective-guided discrete diffusion. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2412.17780.

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

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations (ICLR)*, 2021.

Jiannan Yang, Veronika Thost, Xiang Ling, and Tengfei Ma. Oracle-budgeted molecular optimization with short-term graph memory, 2026. URL https://arxiv.org/abs/2607.28437.

Seokwon Yoon, Youngbin Choi, Seunghyuk Cho, Seungbeom Lee, MoonJeong Park, and Dongwoo Kim. Routing by reaching: Composition of pre-trained GFlowNets for multi-objective generation. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.21565.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):10752, 2019.

APPENDIX

A RELATED WORK

Molecular generation spans latent-variable, graph-to-graph, reinforcement-learning, diffusion, and flow-based approaches, while molecular optimization has also been formulated directly through graph edits, fragment proposals, evolutionary operators, and MCMC transitions (Gómez-Bombarelli et al., 2018; Jin et al., 2018; 2019; Zhou et al., 2019; Jensen, 2019; Xie et al., 2021; Fu et al., 2021; Vignac et al., 2023; Campbell et al., 2024). Recent work has begun to relax the fixed-size assumption of graph generation: GrIDDD introduces node insertion and deletion into discrete graph diffusion, allowing molecular graphs to grow and shrink during denoising (Ninniri et al., 2025). COMPOSE instead learns a goal-independent stochastic law over an explicit, state-dependent molecular rewrite space, making the transition process itself the object of subsequent control.

Fragment-based generative models provide a complementary approach to constrained molecular design. SAFE introduced a fragment-based molecular representation and benchmark spanning motif extension, scaffold decoration, linker design, scaffold morphing, and superstructure generation (Noutahi et al., 2024). GenMol combines this representation with masked discrete diffusion and applies a pretrained model across fragment-constrained generation and molecular optimization (Lee et al., 2025). InVirtuoGen similarly reuses a pretrained fragment-based generator across generation and optimization tasks, with task-specific procedures for conditioning and search. COMPOSE differs in how these constraints enter the model: the learned reference process remains fixed, while structural requirements restrict the executable programs available to the controller.

A related line of work treats generation and adaptation directly through stochastic processes. Generator Matching provides a general framework for learning Markov generators from tractable conditional processes, while Edit Flows develops insertion, deletion, and substitution dynamics for variable-length discrete generation (Holderrieth et al., 2025; Havasi et al., 2025). MadSBM is a close mathematical relative, formulating peptide generation as a controlled Markov process on an amino-acid edit graph, although its learned control constructs the generative transport rather than adapting an already learned molecular process (Goel and Chatterjee, 2026). Value Matching steers a frozen flow model through a learned value function, and pCoMole studies constrained molecular editing with discrete flows (Jensen et al., 2026; Chen et al., 2026). Future-aware generative control is therefore not unique to COMPOSE: value-guided flow adaptation and pCoMole likewise use learned or rollout-estimated future value. Our distinction is the substrate on which that control operates. It is a learned canonical transition law over executable molecular graphs that can also support structural conditioning and feedback-driven optimization without retraining the underlying process.

Multi-objective and constrained molecular design has been addressed through surrogate-based optimization, evolutionary search, and inference-time guidance of generative models. InversionGNN combines multi-property prediction with gradient-based Pareto search in discrete molecular space (Niu et al., 2025), while PepTune and AReUReDi use discrete generative models with multi-objective tree search or Pareto-oriented refinement (Tang et al., 2025; Chen et al., 2025). Routing by Reaching similarly composes pretrained GFlowNets for multi-objective generation at inference time (Yoon et al., 2026). COMPOSE instead performs task-specific control over a reusable molecular transition process, allowing the same learned process to support structural constraints, future-aware editing, and black-box optimization.

B PRELIMINARIES

COMPOSE uses two standard ideas from stochastic control. Structured molecular programs act as temporally extended actions, so controller decisions occur at program boundaries rather than after every primitive rewrite. For similarity-constrained molecular editing, we additionally use the finite-horizon Doob transform to bias a fixed reference process toward desirable terminal states. We briefly review both constructions here.

B.1 TEMPORALLY EXTENDED ACTIONS

Let $R(y \mid x)$ denote a reference transition kernel on a discrete state space. A program o initiated from state x executes a sequence of primitive transitions and terminates after a random duration ℓ in state x'. Its induced transition law can be written as

$$B(x', \ell \mid x, o) = \Pr(X_\ell = x',\, L = \ell \mid X_0 = x, o). \tag{12}$$

Because different programs may require different numbers of primitive transitions, control at program boundaries is naturally semi-Markov. The controller chooses a new program only after the previous one terminates, while the primitive transitions within each program remain part of the underlying execution.

In COMPOSE, structured molecular programs play this role. A program may compose several executable rewrites into a coordinated molecular change, while every intermediate state is still produced through the same molecular executor. Program duration, controller decisions, and expensive oracle evaluations are therefore distinct quantities.

B.2 FINITE-HORIZON DOOB CONTROL

Let $R(y \mid x)$ again denote a reference Markov kernel and let $g_z(x) \geq 0$ assign terminal desirability under goal z. For b transitions remaining, define

$$h_0(x, z) = g_z(x), \qquad h_b(x, z) = \sum_y R(y \mid x)\, h_{b-1}(y, z). \tag{13}$$

Whenever $h_b(x, z) > 0$, the finite-horizon Doob transform gives

$$P^{z,b}(y \mid x) = R(y \mid x) \frac{h_{b-1}(y, z)}{h_b(x, z)}. \tag{14}$$

Starting from x_0 with horizon K, the resulting terminal law is

$$P^z(X_K = y \mid X_0 = x_0) = \frac{P_R(X_K = y \mid X_0 = x_0)\, g_z(y)}{\mathbb{E}_R[g_z(X_K) \mid X_0 = x_0]}. \tag{15}$$

The transform reweights trajectories already supported by the reference process and does not introduce new transitions. In COMPOSE, this construction underlies the future-value controller used for similarity-constrained molecular editing, where a learned model h_ϕ approximates the exact backward value. The structural and feedback-driven controllers used elsewhere in the paper act on the same executable molecular process but do not require the Doob transform.

C THE EXECUTABLE MOLECULAR PROCESS IN FULL

We develop COMPOSE as a reusable stochastic process over executable molecular transformations together with task-specific control over that process. An exact rewrite system first determines which transitions between complete molecular graphs are executable. We then learn a goal-independent reference law over this state-dependent support and push it forward to a canonical molecular successor kernel R_θ. Structured molecular programs compose executable edits into coordinated transformations while preserving the same underlying rewrite semantics. A separate controller then determines how these transformations are used for a downstream design problem. In the settings studied here, control takes the form of structural conditioning, feedback-driven program selection, or finite-horizon future-value guidance, while the executor and learned reference process remain fixed.

C.1 EXECUTABLE MOLECULAR TRANSITIONS

COMPOSE represents every generative state as a complete molecular graph. Let $n_{\max}$ denote the maximum number of heavy atoms admitted by the declared molecular state space. For $n \in \{1, \ldots, n_{\max}\}$, let $\mathcal{X}_n$ be the set of connected molecular graphs with n heavy atoms whose node and edge attributes lie in the declared vocabularies of elements, bond orders, formal charges, and implicit hydrogens, and in which every atom satisfies its declared valence constraints. The molecular state space is

$$\mathcal{X} = \bigsqcup_{n=1}^{n_{\max}} \mathcal{X}_n. \tag{16}$$

Thus each state $x \in \mathcal{X}$ is a complete molecule rather than a masked, corrupted, or partially specified representation.

Molecular transitions are generated by typed graph rewrites. An edit mark a specifies a rewrite type together with its molecular match and typed payload, and $\mathsf{T}(x, a)$ denotes the graph obtained by applying that rewrite to x. The full atom-, bond-, and ring-level edit language is specified in Section E.1. From a state x, the executor exposes only the finite, state-dependent set

$$\mathcal{A}(x) = \big\{a : \ a \text{ matches } x \text{ and } \mathsf{T}(x, a) \text{ passes all executor guards}\big\}. \tag{17}$$

Execution follows a propose-validate-commit rule: a successor is admitted only if its attributes remain in the declared vocabularies, affected valences are satisfied, the graph remains connected, and its heavy-atom count lies in $\{1, \ldots, n_{\max}\}$. An edit that fails any guard contributes no transition to $\mathcal{A}(x)$. Inadmissible molecular states are therefore absent from the stochastic support rather than discouraged through a learned penalty.

The decomposition of $\mathcal{X}$ by heavy-atom count makes the process trans-dimensional. At the primitive level, an admitted rewrite changes the heavy-atom count by at most one, giving *birth*, *death*, or *same-cardinality* transitions between $\mathcal{X}_n$ and $\mathcal{X}_{n+1}$, $\mathcal{X}_{n-1}$, or $\mathcal{X}_n$, respectively. Birth and death rewrites create or remove typed graph entities that participate in valence, connectivity, and ring perception rather than merely activating or deactivating padded coordinates. A trajectory may therefore grow, shrink, and restructure a molecular graph within a common state space.

By construction,

$$a \in \mathcal{A}(x) \quad \Longrightarrow \quad \mathsf{T}(x, a) \in \mathcal{X}. \tag{18}$$

Consequently, any sequence of admitted primitive rewrites remains in the declared molecular state space. Structured molecular programs introduced below compose these same primitive transitions into larger coordinated transformations; they do not bypass the executor or introduce a separate notion of molecular validity.

The legal edit sets $\mathcal{A}(x)$ therefore determine which molecular transitions are possible, but not how probability should be distributed among them. We next learn a goal-independent stochastic law over this executable support.

C.2 Learned molecular reference process

Having defined which molecular rewrites are executable, we next learn how probability is distributed over them. For every state x with $\mathcal{A}(x) \neq \emptyset$, the reference model defines a normalized conditional law over executable edit marks,

$$p_\theta(a \mid x), \qquad \sum_{a \in \mathcal{A}(x)} p_\theta(a \mid x) = 1. \tag{19}$$

Writing an edit as $a = (k, o, \eta)$, with rewrite family k, matched molecular operands o, and typed payload η, we parameterize

$$p_\theta(a \mid x) = p_\theta(k \mid x)\, p_\theta(o, \eta \mid k, x). \tag{20}$$

A shared molecular encoder represents the current graph, while family-specific scorers normalize only over candidates admitted by the executor.

The reference model receives no downstream objective, task budget, or task-specific constraint. Structural requirements, optimization objectives, and observed task feedback enter only through the controllers introduced later. Thus p_θ learns a task-independent distribution over molecular transformations supported by the executor, and its parameters remain fixed across the downstream settings studied in this paper.

Training supervision consists of executable teacher transitions. Teacher programs may be constructed using source and target structures, but every stored transition is replayed through the same executor used at inference, and neither the target structure nor the teacher program is provided to the reference model when predicting from x. Because multiple executable marks may lead to the same molecular graph, we train on the resulting molecular successor rather than on a particular syntactic realization of that transition.

Let $R_\theta(y \mid x)$ denote the induced probability of the distinct committed molecular successor y, defined formally in Section C.3. For teacher transitions $x_i \to y_i^\star$, we minimize the canonical-successor negative log-likelihood

$$\mathcal{L}_{\mathrm{ref}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log R_\theta(y_i^\star \mid x_i). \tag{21}$$

The model therefore learns which executable molecular successor follows a given state under the training transition distribution, rather than predicting a downstream property or optimizing a task-specific objective.

The learned mark law is not yet the molecular transition kernel used by the controllers: distinct executable marks can produce the same molecular state. We therefore push p_θ through the executor and aggregate probability by canonical molecular outcome.

C.3 Canonical molecular successors

We now push the learned edit law through the executor to obtain a stochastic process over distinct molecular states. Multiple executable marks can produce the same molecular graph because of graph symmetries, equivalent matches, or alternative internal rewrite descriptions. For a successor y, define

$$\mathcal{G}_y(x) = \big\{a \in \mathcal{A}(x) : \mathsf{T}(x, a) \simeq y\big\}, \tag{22}$$

where $\simeq$ denotes equivalence under the registered molecular canonicalization. The induced probability mass of y is

$$\bar{p}_\theta(y \mid x) = \sum_{a \in \mathcal{G}_y(x)} p_\theta(a \mid x). \tag{23}$$

Thus probability assigned to a molecular transition is the aggregate mass of every executable mark that realizes it. For example, deleting any of the three symmetry-equivalent terminal methyl carbons in 2-methylpropane, `CC(C)C`, produces propane; canonicalization exposes propane once with the combined probability of those marks rather than overweighting it because of representational multiplicity.

Let $\mathcal{S}(x)$ denote the distinct non-self canonical molecular successors of x. Conditioning the pushforward distribution on a committed molecular state change defines the reference successor kernel

$$R_\theta(y \mid x) = \frac{\mathbf{1}\{y \in \mathcal{S}(x)\}\, \bar{p}_\theta(y \mid x)}{\sum_{y' \in \mathcal{S}(x)} \bar{p}_\theta(y' \mid x)}. \tag{24}$$

If $\mathcal{S}(x) = \emptyset$, x is terminal for the primitive reference process. Otherwise, $R_\theta(\cdot \mid x)$ is a normalized distribution over distinct executable molecular successors. Iterating this kernel defines the goal-independent molecular process learned in Section C.2. Structured programs introduced below compose transitions from this same executable process rather than defining a separate molecular state space or transition semantics.

Proposition 1 (Canonical normalization)**.** *For every nonterminal state x,*

$$\sum_{y \in \mathcal{S}(x)} R_\theta(y \mid x) = 1.$$

Moreover, any refinement or coarsening of the internal mark representation that preserves the aggregate probability mass of every canonical successor set leaves R_θ unchanged. Consequently, state-level control defined on canonical molecular successors is invariant to such representational refinements.

This invariance depends on aggregating probability by molecular outcome before state-level control is applied: reweighting or truncating individual marks can otherwise reintroduce dependence on arbitrary edit multiplicities. We next use this canonical molecular process as the substrate for structured molecular programs and task-specific control.

C.4 STRUCTURED MOLECULAR PROGRAMS

Primitive rewrites define the support of the molecular process, but many useful molecular transformations require several dependent edits. We therefore compose primitive rewrites into structured molecular programs. A program

$$\pi = (a_1, \dots, a_L) \tag{25}$$

acts on an initial state x through

$$x^{(0)} = x, \qquad x^{(\ell)} = \mathsf{T}\Big(x^{(\ell-1)}, a_\ell\Big), \qquad \ell = 1, \dots, L. \tag{26}$$

The program is executable from x if

$$a_\ell \in \mathcal{A}\Big(x^{(\ell-1)}\Big) \qquad \text{for every } \ell = 1, \dots, L. \tag{27}$$

We write $\Pi(x)$ for the set of executable programs from x and define the program endpoint

$$\mathsf{T}_\pi(x) = x^{(L)}. \tag{28}$$

Programs do not enlarge the primitive molecular state space or bypass the rewrite system. By Equation (18), every executable step satisfies $x^{(\ell)} \in \mathcal{X}$, so every prefix of an executable program is itself a sequence of valid molecular states. A program therefore specifies a path through the same executable molecular process rather than a direct jump between its initial and terminal structures.

The distinction matters because many coordinated molecular transformations cannot be represented by choosing each primitive edit independently. A later rewrite may act on an atom, bond, or ring created by an earlier rewrite, and a useful endpoint may require passing through several intermediate molecular states before the coordinated transformation is complete. Ring construction, region replacement, and other multi-edit transformations can therefore be expressed as programs while retaining the same primitive executor semantics.

Programs provide the higher-level action space used by the controllers below. A controller may restrict which programs are admissible, construct programs from particular transformation families, or use the reference process R_θ when proposing and ranking their constituent transitions. These choices do not modify the executor or the learned reference process. They determine how executable primitive transitions are composed and selected for a particular design problem.

C.5 CONTROL OF THE FROZEN MOLECULAR PROCESS

Once the executable process and reference law have been learned, downstream task information enters only through control. At a decision point, let x denote the current molecule, c the task specification, and $\mathcal{H}$ any feedback collected during the current search. Depending on the setting, the controller acts either on canonical molecular successors $y \in \mathcal{S}(x)$ or on executable programs $\pi \in \Pi(x)$. We write the corresponding controlled law abstractly as

$$Q_c(u \mid x, \mathcal{H}), \qquad u \in \mathcal{U}(x), \qquad \mathcal{U}(x) \in \{\mathcal{S}(x), \Pi(x)\}. \tag{29}$$

The controller may restrict which choices are admissible or change how probability and computation are distributed among them, but it cannot introduce a molecular transition outside the executable process defined above.

The form of c and $\mathcal{H}$ depends on the design problem. For fragment-constrained generation, structural requirements restrict the executable programs available from the current molecule. For black-box optimization, the controller adapts program selection from objective values observed during the search. For similarity-constrained molecular editing, a learned future value reweights the primitive reference kernel toward successors with desirable remaining continuations. In all three cases, the executor and R_θ remain fixed.

This separation allows the same learned molecular process to support different forms of molecular design without retraining the underlying transition model. We describe each form of control below.

D THEORETICAL PROPERTIES

We record the two theoretical properties used by the controllers in the main text. The first concerns invariance to the internal representation of executable edit marks. The second, given in the following subsection, characterizes exact finite-horizon control of the reference process.

D.1 INVARIANCE OF CANONICAL SUCCESSOR CONTROL

The executor may admit several edit marks that produce the same canonical molecular successor. The construction of R_θ in Section C.3 removes this representational multiplicity by aggregating probability before control is applied. The following proposition makes the resulting invariance precise.

Definition 2 (Successor-mass-preserving re-encoding)**.** Consider two executable mark representations $(\mathcal{A}, p_\theta, \mathsf{T})$ and $(\mathcal{A}', p'_\theta, \mathsf{T}')$ of the same molecular state space. They are *successor-mass preserving* if they induce the same committed canonical successor set $\mathcal{S}(x)$ and, for every x and $y \in \mathcal{S}(x)$,

$$\sum_{\substack{a \in \mathcal{A}(x) \\ \mathsf{T}(x,a) \simeq y}} p_\theta(a \mid x) = \sum_{\substack{a' \in \mathcal{A}'(x) \\ \mathsf{T}'(x,a') \simeq y}} p'_\theta(a' \mid x). \tag{30}$$

Proposition 3 (Canonical successor invariance)**.** *Under a successor-mass-preserving re-encoding, the canonical reference kernel $R_\theta(y \mid x)$ is unchanged. Consequently, any successor-level controller of the form*

$$K_\Psi(y \mid x) = \frac{R_\theta(y \mid x)\, \Psi(x, y)}{\sum_{y'} R_\theta(y' \mid x)\, \Psi(x, y')}, \qquad \Psi(x, y) \geq 0, \tag{31}$$

is unchanged wherever the denominator is positive, provided Ψ depends only on canonical molecular states.

Proof. Successor-mass preservation gives the same aggregate probability for every canonical successor y. Summing these masses over the common successor set therefore gives the same normalization constant, so $R_\theta(y \mid x)$ is identical under both representations. Because Ψ depends only on canonical states, both the numerator and denominator of Equation (31) are then unchanged. □

Canonicalization must precede nonlinear state-level control for this invariance to hold. For example, suppose a successor has total mark probability p. Applying the nonlinear transformation $u \mapsto u^\beta$ at the mark level gives mass p^β if the successor is represented by one mark, but $m(p/m)^\beta$ if the same mass is split evenly across m equivalent marks. These quantities differ unless $\beta = 1$. Aggregating by molecular outcome before control prevents such arbitrary mark multiplicities from affecting the controlled process.

D.2 EXACT FINITE-HORIZON CONTROL

We next state the exact finite-horizon result underlying the future-value controller in Section 2.7. To account for trajectories that terminate before the declared horizon, we adjoin an absorbing cemetery state ∂ and set $g_z(\partial) = 0$.

Theorem 4 (Finite-horizon control)**.** *Let R_θ be the canonical reference kernel and let $g_z : \mathcal{X} \cup \{\partial\} \to \mathbb{R}_{\geq 0}$ be finite-valued, with $g_z(\partial) = 0$. Define*

$$h_0(x, z) = g_z(x), \qquad h_b(x, z) = \sum_y R_\theta(y \mid x)\, h_{b-1}(y, z), \tag{32}$$

where terminal molecular states transition to ∂ with probability one and ∂ is absorbing. Suppose $0 < h_K(x_0, z) < \infty$. Wherever $h_b(x, z) > 0$, define

$$P^{z,b}(y \mid x) = R_\theta(y \mid x) \frac{h_{b-1}(y, z)}{h_b(x, z)}. \tag{33}$$

Then $P^{z,b}(\cdot \mid x)$ is normalized and has support contained in that of R_θ. Applying the controlled kernels with remaining horizons $K, K-1, \ldots, 1$ gives

$$\Pr_{P^z}(X_K = y \mid X_0 = x_0) = \Pr_{R_\theta}(X_K = y \mid X_0 = x_0) \frac{g_z(y)}{h_K(x_0, z)}. \tag{34}$$

Proof. For any state with $h_b(x, z) > 0$,

$$\sum_y P^{z,b}(y \mid x) = \frac{\sum_y R_\theta(y \mid x) h_{b-1}(y, z)}{h_b(x, z)} = 1 \tag{35}$$

by the backward recursion. Moreover, $R_\theta(y \mid x) = 0$ implies $P^{z,b}(y \mid x) = 0$, so control introduces no transition outside the reference support.

For any length-K path $x_{0:K}$ with positive controlled probability,

$$\Pr_{P^z}(x_{1:K} \mid x_0) = \prod_{j=0}^{K-1} R_\theta(x_{j+1} \mid x_j) \frac{h_{K-j-1}(x_{j+1}, z)}{h_{K-j}(x_j, z)} \tag{36}$$

$$= \Pr_{R_\theta}(x_{1:K} \mid x_0) \frac{h_0(x_K, z)}{h_K(x_0, z)} \tag{37}$$

$$= \Pr_{R_\theta}(x_{1:K} \mid x_0) \frac{g_z(x_K)}{h_K(x_0, z)}, \tag{38}$$

because the intermediate h-terms telescope. Summing over all reference paths ending at y gives Equation (34). □

The theorem is specific to the declared horizon. At an intermediate time j, the controlled marginal is proportional to

$$\Pr_{R_\theta}(X_j = x \mid X_0 = x_0)\, h_{K-j}(x, z), \tag{39}$$

rather than directly to $\Pr_{R_\theta}(X_j = x \mid X_0 = x_0) g_z(x)$. Thus the controller favors states with desirable remaining continuations, not simply states with high immediate terminal desirability.

D.3 Learned future values and twisted sequential Monte Carlo

Exact backward values are intractable in the molecular state space, so we learn a nonnegative approximation

$$h_\phi(x, z, b) \approx h_b(x, z) \tag{40}$$

from finite-horizon continuations of the frozen reference process. This model estimates the desirability of the terminal states reachable from x with b transitions remaining rather than the immediate desirability of x itself.

Given h_ϕ, a locally twisted successor distribution is

$$Q_\phi^{z,b}(y \mid x) = \frac{R_\theta(y \mid x)\, h_\phi(y, z, b-1)}{\sum_{y'} R_\theta(y' \mid x)\, h_\phi(y', z, b-1)}. \tag{41}$$

Whenever the denominator is positive, this is a normalized distribution whose support is contained in that of R_θ. Thus future-value guidance can redistribute probability among executable molecular successors but cannot introduce a transition unavailable to the frozen reference process.

Unlike the exact construction in Theorem 4, however, replacing h_b by h_ϕ does not in general recover the exact terminal g_z-tilted distribution. We therefore use h_ϕ as a twisting function inside sequential Monte Carlo rather than treating Equation (41) as an exact Doob transform. Particles evolve through successive executable transitions of the frozen reference process, while the future-value model directs computation toward trajectories whose continuations are predicted to reach desirable terminal states. Particle weights are updated under this twist and trajectories are periodically resampled, concentrating computation on promising continuations without changing the underlying molecular transition support.

For similarity-constrained molecular editing, the goal z specifies the target property region together with the source-relative similarity requirement. Independent controlled trajectories are initialized from the same source molecule and propagated for the fixed editing horizon. The returned molecules are the terminal states of these controlled trajectories rather than the highest-scoring members of a pooled endpoint search. The executor and R_θ remain fixed throughout; only the future-value model and resulting particle allocation depend on the editing objective.

E IMPLEMENTATION DETAILS

E.1 PRIMITIVE REWRITE VOCABULARY

The executable process uses eight primitive rewrite families. Each rewrite is proposed on the current molecular graph and committed only if its successor passes the executor guards described in Section C.1. These primitives define the molecular transition support used throughout the paper; structured programs compose them but do not introduce additional primitive transition semantics.

The paper-facing names above describe executable rewrite semantics. In the implementation, `cycle_close`, `cycle_open`, and `ring_aromaticity_restate` are recorded under the model-family identifiers `cycle_insert`, `cycle_attach`, and `ring_system_restate`, respectively; the remaining family names agree across the two conventions.

Whole-ring construction, deletion, and larger region transformations are not additional primitive rewrites. They are realized by structured programs that compose the operations above, so every intermediate molecular state remains subject to the same executor and validity checks.

E.2 REFERENCE PROCESS ARCHITECTURE AND TRAINING

The reference process is trained once and then frozen for every downstream experiment. A shared molecular encoder maps the current graph to a 256-dimensional state representation. A family head scores the available rewrite families, and family-specific heads score the legal operands and typed payloads within each family. All normalization is performed only over candidates admitted by the executor, so the learned model assigns probability only to transitions in $\mathcal{A}(x)$.

The implementation also provides the encoder with the progress coordinate used by the reference-process training scheme. This coordinate is part of the fixed reference model and is distinct from the downstream search budget; we suppress it in the main notation for clarity. The reference model does not receive the starting lead, downstream objective, task constraints, oracle values, or remaining optimization budget.

We train the model on executable teacher transitions using the canonical-successor likelihood in Equation (21). For the checkpoint used in this paper, teacher and importance weights are unit-valued, while the corpus sampler determines the training measure through its fixed distribution over rewrite families. Terminal rows are excluded because they contain no committed successor. Model architecture, optimization schedule, checkpoint selection, and all training hyperparameters are fixed independently of the downstream evaluation panels. The resulting checkpoint is used without retraining or fine-tuning in all experiments.

E.3 STRUCTURAL PROGRAM CONSTRUCTION

Fragment-constrained tasks specify molecular structure that must be retained together with the interfaces through which new structure may be introduced. We represent this information as a structural specification c_{str} containing the protected molecular entities, the permitted attachment interfaces, and the task-specific endpoint conditions. The specification restricts program construction but does not modify the executor or the learned reference process.

For a molecule x, let $\Pi(x)$ denote the executable programs defined in Section C.4. We restrict this set to

$$\Pi_{c_{\mathrm{str}}}(x) = \{\pi \in \Pi(x) : \pi \text{ preserves the protected structure and satisfies the prescribed interfaces}\}. \tag{42}$$

Preservation is enforced during program execution: a primitive rewrite that would delete or modify a protected graph entity is unavailable to the program. Attachment choices are likewise restricted to the interfaces allowed by c_{str}. Every admitted step is still executed through the primitive rewrite system, so intermediate states remain valid molecular states and larger structural changes inherit the same executor semantics.

Different fragment tasks correspond to different structural specifications rather than different molecular generators. Motif extension and scaffold decoration expose selected interfaces around retained structure, while linker design and scaffold morphing constrain generation between fixed molecular

Table 4: **Primitive rewrite vocabulary.** Each family is an executable graph rewrite with state-dependent structural and chemical applicability conditions.

Rewrite	Cardinality	Applicability and effect
`atom_insert`	birth	Add an allowed atom state, either from the null source or bonded to one existing atom with compatible bond order and valence.
`atom_delete`	death	Remove an atom when the remaining graph stays connected and the resulting hydrogen and valence states are valid.
`atom_restate`	same	Change the supported element, valence, or implicit-hydrogen state of an existing atom while preserving a valid successor.
`bond_reorder`	same	Change the order of an existing bond together with any required hydrogen updates.
`bond_reroute`	same	Detach a pendant fragment across a bridge edge and reconnect it at a different valid attachment site in one committed transaction.
`cycle_close`	same	Insert an absent bond between two existing atoms with sufficient valence headroom, increasing graph cycle rank by one.
`cycle_open`	same	Remove a non-bridge cycle edge with the required hydrogen compensation, decreasing graph cycle rank by one.
`ring_aromaticity_restate`	same	Apply a coordinated set of bond-order changes to a matched cyclic system, subject to the same executor validation.

boundaries. Superstructure generation instead requires the designated structure to remain present while additional structure is introduced. The same frozen R_θ and primitive rewrite vocabulary are used in each case.

The completed molecule is finally checked against the benchmark-specific containment and connectivity conditions. This endpoint check is distinct from chemical validity: the executor guarantees that committed states remain valid molecules, whereas the structural evaluator determines whether a valid molecule also satisfies the requested fragment constraint.

E.4 Feedback-driven program control

For black-box optimization, we maintain a task-specific archive of scored molecular endpoints and the exact programs that produced them. Each archive record contains the source state, program, binding, executed intermediate states, canonical endpoint, objective protocol, and charged evaluation outcome. Scored endpoints can subsequently serve as parents for further programs.

At each search round, the controller selects archive parents and proposes complete programs through the proposal channels declared by the experiment. The program compiler and executor establish the exact endpoint before an objective evaluation. We apply inexpensive endpoint requirements and canonical deduplication at this stage, record failed or ineligible proposals, and lock the selected query batch before making objective calls. Evaluated outcomes then update the archive. This ordering separates proposal work, executable molecular changes, endpoint eligibility, and charged objective evaluations.

The proposal law and feedback allocation are fixed by each experiment's configuration. In particular, the frozen T4 program-only benchmark selects archive parents without score ranking, tempers repeatedly exhausted parents, and does not fit a program-value model. Its scored outcomes extend the archive and determine the reported incumbent, but they do not turn the proposal law into a sample from R_θ or an exact Doob transform. Other developmental controller variants may fit an improvement model from charged observations; those variants require separate protocol and result identification. In every variant, the executor and reference-model parameters remain unchanged.

E.5 FUTURE-VALUE MODEL TRAINING

For similarity-constrained editing, we trained the region-goal future-value head on finite-horizon continuations of the frozen reference process. Its input combines a molecular graph embedding, goal coordinates, signed property margins, and a one-hot encoding of the remaining primitive-transition budget. The head estimates the probability that the state at the declared terminal horizon belongs to the specified region. The architecture and training settings are given in Table 5.

We split by source molecule before fitting so states from one reference trajectory cannot appear in both training and validation. Input normalization is fitted on training states and reused for validation. The terminal boundary $h_0 = g_z$ is imposed directly. We selected the head checkpoint by held-source validation loss, leaving the R_θ encoder and transition parameters unchanged. Goal coordinates are continuous inputs, so the head can be evaluated at threshold pairs outside the discrete training grid; such evaluation does not by itself establish calibrated reachability at those pairs.

F DATA AND TRANSITION SUPERVISION

F.1 MOLECULAR SOURCES AND SPLIT

The editing corpus was compiled from a fixed 500,000-molecule GuacaMol source pool into executable local transitions, real-endpoint analogue relations, and reversible synthetic walks. The learned object is a conditional successor law, not an unconditional molecular density. The frozen training source universe contains 96,094 molecules. Evaluation sources were assigned before model fitting, with role precedence final test, controller validation, validation, then training. Exact duplicate sources were removed from lower-precedence roles. The matched reserve was made cluster-disjoint from training using an ECFP4 Tanimoto threshold of 0.95 with formula and scaffold blocking.

The compiled corpus is identified by its recorded dataset, library, packed-store, and reserve digests. The receipts do not bind the original raw asset end to end, so raw-source provenance is less complete than compiled-corpus identity. A source-statistics sample of 4,000 training molecules had median 26 heavy atoms, molecular weight 371.6, and three rings. These measurements describe the training density and do not narrow the executor's declared molecular support.

F.2 EXECUTABLE TRANSITION CONSTRUCTION

All 1,803,032 stored training transitions passed the frozen structural gate. The ring-family supervision is synthetic: the training-chunk ledger attributes 122,183 `cycle_insert`, 84,940 `cycle_attach`, and 16,380 `ring_system_restate` transitions entirely to reversible synthetic walks, with none in the four data-backed lanes. The pinned distribution-matched subset contains 23,100 synthetic-walk transitions among 145,756 transitions. The analogue lane contains 344,153 molecular-matched-pair records; its upstream molecular-source chain is not fully documented.

Consequently, ring-family held-out likelihood evaluates recovery of legal edits under synthetic perturbation. It does not establish a learned distribution of observed ring transformations or scaffold hopping in medicinal chemistry. We report the data-backed and synthetic-only evaluation aggregates separately in Section G.1.

F.3 EVALUATION PROVENANCE

The reference checkpoint was frozen across the reported experiments. Source roles and panel membership were fixed before fitting, and the evaluation panels used for the reference-process study are exact- and near-duplicate-disjoint from training under the declared checks. The official QED cohort is a separate benchmark population. A pre-freeze smoke test executed one official source; its output was not inspected and is excluded from all analyses. The sealed 33,683-record final-test role was not used for any result reported here.

Table 5: **Region-goal future-value model.** Settings used for the similarity-constrained editing controller.

Component	Setting
Input width	1055
Head	MLP $1055 \to 512 \to 256 \to 128 \to 1$, ReLU, dropout 0.1
Output	Sigmoid reachability
Loss	Binary cross-entropy $+ 0.3\times$ Bellman consistency
Optimizer	Adam, learning rate 10^{-3}
Goal grid	5 QED thresholds $\times$ 4 similarity floors
Validation split	0.15, held out by source molecule

G Experimental details

We report protocols and supporting measurements for the current experiments in the order used in the main paper. Developmental trajectory-control panels are collected separately in Section H; they do not establish the headline benchmark comparisons.

G.1 Reference-law evaluation

We evaluate the frozen reference law on held-out transitions from the matched reserve by measuring the probability assigned to the realized canonical successor. The uniform, empirical-family and learned laws are evaluated on the identical executable support and canonical successor quotient; only their probability assignments differ. The empirical-family law uses corpus-level edit-family frequencies renormalized over the families available at the current state and distributes each family's mass uniformly across its available successors. Two intermediate arms decompose the comparison: an empirical family with learned identity, which keeps corpus family frequencies and learns only which successor within the chosen family is realized, and a learned family with uniform identity, which learns only which family fires. Together they separate choosing which edit to make from choosing where to apply it.

The primary quantity is the mean negative log-probability of the realized successor, with the scored transition as the unit and paired differences by construction, since every law scores the same transition. Cells below a preregistered minimum size of 20 are not reported separately; no scored transition was skipped and no partition failed to compile. Performance against the uniform law tests whether learning improves on executable support alone, and performance against the empirical-family law tests whether the reference law captures state-dependent preference beyond global family frequency. Neither comparison is interpreted in terms of physical molecular kinetics or synthetic-route likelihood.

The decomposition is strongly asymmetric. Learning only which family fires recovers little, improving on corpus family frequencies by 0.20 nats overall, whereas learning only where within a family the edit lands recovers most of the effect at 4.10; the full model adds a further 0.20 beyond that. In the terms used in Section G.1, the empirical-family-to-full improvement is 1.4616 nats and the identity-only arm recovers 1.2674 of it, or 86.7%, against 13.3% for the family gate alone. The reference law's aggregate advantage is therefore driven primarily by state-dependent successor identity rather than by the family gate. The empirical-family control is also not uniformly strong: on both `cycle_insert` cells, on `atom_insert` and on `atom_restate` it is worse than assigning equal probability to every distinct successor, so on birth and insertion families a corpus frequency prior misallocates mass relative to ignorance. On four cells the empirical-family/learned-identity ablation assigns higher probability to the realized successor than the full model, indicating that the learned family gate does not uniformly improve over corpus-level family frequencies and that the identity model carries the result on those families.

Aggregating by supervision provenance sharpens this. On the $2{,}744$ transitions from families with data-backed supervision, the reference law retains a 1.06-nat advantage over empirical family frequencies (5.55 against 4.49), so the main-text claim survives the obvious confound. On that subset, however, the empirical-family/learned-identity ablation reaches 4.25 nats and therefore assigns higher probability to the realized successor than the full model does, and the learned-family/uniform-identity arm at 5.79 is worse than corpus family frequency. The learned family gate is thus not uniformly beneficial, and on data-backed families the identity model carries the result in spite of it. The complementary synthetic-

Capability cell	n	unif.	emp. fam.	*emp+id*	*fam+unif*	R_θ
`atom_restate`: element identity change	725	6.21	6.54	4.84	7.28	5.58
`atom_delete`: leaf death	601	6.20	3.36	2.68	3.69	3.01
`atom_insert`: one-neighbor birth	572	6.14	7.12	5.23	6.97	5.09
`bond_reroute`: single-atom pendant, cyclic	464	6.24	5.26	4.07	5.48	4.29
`bond_reroute`: multi-atom pendant, cyclic	216	6.32	5.56	4.38	6.14	4.96
`ring_system_restate`: dearomatization†	202	6.25	2.84	2.97	1.72	1.85
`ring_system_restate`: aromatization†	165	6.51	2.85	2.34	1.12	0.61
`cycle_insert`: close to monocyclic†	138	6.47	7.19	3.27	4.80	0.88
`cycle_attach`: open from monocyclic†	104	6.16	5.24	5.20	3.34	3.29
`bond_reorder`: bond-order decrease	83	6.32	4.30	4.01	3.18	2.89
`bond_reorder`: bond-order increase	83	6.49	4.71	4.40	3.29	2.99
`cycle_insert`: close to polycyclic†	81	6.55	7.37	3.95	5.26	1.84
`cycle_attach`: open from polycyclic†	69	6.42	5.82	5.71	4.08	3.96
Overall	3545	6.25	5.37	4.10	5.17	**3.90**

Table 6: Negative log-probability of the realized canonical successor by capability cell, with lower values indicating better performance. All five laws are evaluated on the same executable support and canonical quotient. Only the probability assignment differs. The *emp+id* arm combines empirical family frequencies with learned successor identity. The *fam+unif* arm combines learned family probabilities with uniform successor identity. Cells below the preregistered minimum size of 20 are not reported separately. No scored transition was skipped and no partition failed to compile. †Supervision for these cells comes entirely from the synthetic-perturbation lane, as detailed in the provenance note below.

only subset comprises 759 transitions, where the reference law reaches 1.79 nats against 4.72 for empirical family frequencies; the aggregate improvement reported in Section G.1 is correspondingly larger than the data-backed figure. The partition is computed from the emitted family-lane provenance artifact rather than from a hand-specified family list, and the restricted aggregates are reproduced from it.

The cells in which the reference law performs best are those whose supervision is synthetic, as recorded in Section F. On the ring and cycle families the model recovers a perturbation of a kind it was trained to invert, and the aggregate improvement should not be read as evidence about molecular kinetics. Resolving instead by the number of available successors gives 4.86 nats for states with no alternative in the recorded band and 5.51, 5.44 and 5.51 for successor counts of 1-4, 5-24 and 25 or more under the empirical-family law, against 4.34, 4.03, 4.00 and 4.21 for the learned-identity arm, so the learned advantage is not an artifact of nearly deterministic rows.

G.2 Fragment-constrained generation

We use the ten drug-derived prompts for each task in the SAFE fragment benchmark. Each task specifies a molecular structure that must be preserved together with task-specific attachment or connectivity requirements. Across the fragment-constrained tasks, COMPOSE uses the same learned reference and primitive executor, while task-specific structural control adapts the shared molecular framework to each constraint. Under the benchmark convention, scaffold morphing reuses the linker prompts and generated molecules, so it is not an independent sampling experiment.

For the headline metrics, each attempted generation occupies one output slot. A no-output attempt remains in the denominator as an unparseable placeholder; it is not silently retried or removed. We pass canonical, connected SMILES to the pinned InVirtuoGen `evaluate_smiles` implementation, which computes chemical validity, uniqueness, quality, and diversity. The canonical-output check is essential because the evaluator deduplicates emitted strings. We report benchmark prompt fidelity separately from chemical validity: an emitted valid molecule that misses the required fragment remains in the headline metric population and is counted as a fidelity failure in the separate check.

The COMPOSE values in Table 1 use the same four headline metric definitions as the published comparison. Distance is omitted because the released IVG headline evaluator does not return it; any supplemental distance must specify the exact reference structure used for fingerprint comparison.

G.3 REFERENCE-PROBABILITY ABLATION IN SUPERSTRUCTURE GENERATION

We tested the learned edit-selection probabilities in the completed superstructure sampler. The comparison law samples uniformly among available rewrite families and then uniformly among masked native mark coordinates in the selected family. Both arms use the same native action masks, structural controls, learned hazard, stopping rules, executor, and benchmark evaluator. The comparison is not uniform over distinct canonical successors.

Across the ten prompts, the mean quality difference was 13.63 percentage points. A percentile bootstrap that resampled prompts after averaging the three seeds within each prompt gave a 95% interval of 4.27 to 23.07 points. The corresponding intervals were 1.07 to 11.70 points for uniqueness and -0.00052 to 0.03111 for diversity. Generated molecules within a prompt were not treated as independent structural problems. The learned law improved quality on six prompts, trailed on two, and tied on two (Table 8). Seed labels match between arms, but their random draws do not pair molecular trajectories.

The comparison isolates learned edit-selection probabilities in the superstructure primitive sampler. The learned hazard remains active in both arms. It does not test the complete reference against a wholly unlearned generator or establish an effect for the finite-panel program controllers used in other fragment tasks.

G.4 SIMILARITY-CONSTRAINED QED EDITING

Each COMPOSE candidate is the terminal molecule of one controlled SMC trajectory; intermediate molecules are not pooled or ranked as alternative returns. The reference law is unchanged from Section G.1, and the future-value controller is trained on sources disjoint from the evaluation panel before that panel is opened.

A receding one-step terminal tilt of the frozen reference law by the immediate property value of each successor was preregistered as the inference policy and refuted on the disjoint development panel, with 0 successes in 320 trajectories, before any official-cohort outcome existed. It is retained as a developmental negative control. What this licenses is narrow: the finite-horizon controller is a pre-outcome amendment consistent with the paper's construction, motivated by a developmental failure of a cheap one-step approximation. It is not a restoration of an originally frozen policy, and compute cost was not the only consideration separating the two.

On the official cohort, COMPOSE solves 446 of the 800 sources at $K = 8$ (55.8%, 95% Wilson interval $[52.3, 59.2]$), against 45.1% reported by GrIDDD at $K = 20$. Two sources did not return a completed shard and are counted as failures, so the denominator is the full benchmark.

G.5 PMO-1K BLACK-BOX OPTIMIZATION

The Practical Molecular Optimization comparison uses 22 objectives under a 1,000-evaluation budget per objective and seed. All objectives start from the same task-independent initialization bank. The objective function is queried only through the benchmark interface, and each returned score is associated with the exact canonical molecule that consumed an evaluation. No target score is used to train or fine-tune the frozen reference model. Program construction, candidate deduplication, and archive updates follow the feedback-driven controller described in Section E.4.

For each objective, the reported statistic is the final top-10 score after the budget is exhausted. The table reports the mean and standard deviation over three seeds and reduces across the same 22-objective set used by the cited no-memory comparator, excluding valsartan SMARTS. A candidate that fails to execute or duplicates an already scored molecule does not become a new scored endpoint; its proposal cost remains part of the run record. This distinction between proposal work and oracle evaluations is needed to interpret efficiency and should be reported alongside the fixed oracle budget.

G.6 T4 SIMILARITY-CONSTRAINED LEAD OPTIMIZATION

Panel and constraints. The panel is five protein targets (PARP1, FA7, 5HT1B, BRAF, JAK2), three seed molecules per target, and two similarity thresholds $\delta \in \{0.4, 0.6\}$, giving 30 cells. A returned molecule is feasible when QED ≥ 0.6, SA ≤ 4, and $\mathrm{Tanimoto}(x, x_0) \geq \delta$ against the cell's seed,

Table 7: **Learned edit-selection probabilities in superstructure generation.** Each arm contains 100 attempted outputs for each of ten prompts and three seeds. Quality, uniqueness, and chemical validity are percentages. Diversity is a fraction. Prompt compliance counts completed molecules satisfying the supplied structural condition. The final row is learned minus uniform.

Selection law	Quality ↑	Uniqueness ↑	Diversity ↑	Validity ↑	Prompt compliance
Learned family and mark	39.033	97.333	0.725074	100.0	3,000/3,000
Uniform family and mark	25.400	90.167	0.708503	100.0	3,000/3,000
Difference	+13.633	+7.167	+0.016571	0	0

Table 8: **Superstructure quality by structural prompt.** Each value averages three seeds of 100 attempts for the indicated prompt. Differences are learned minus uniform in percentage points.

Prompt	Learned	Uniform	Difference
Baricitinib	46.67	25.67	+21.00
Cyclothiazide	0.00	0.00	0.00
Eliglustat	66.67	41.67	+25.00
Erlotinib	37.00	43.67	-6.67
Futibatinib	43.00	10.67	+32.33
Lesinurad	49.33	36.00	+13.33
Liothyronine	44.00	50.33	-6.33
Lovastatin	0.00	0.00	0.00
Maribavir	73.33	36.33	+37.00
Spirapril	30.33	9.67	+20.67

with similarity on Morgan fingerprints of radius two. Among feasible molecules the objective is the QuickVina2 docking score, lower being better. These are endpoint constraints only: intermediate molecular states are not required to satisfy them, which is what makes the trajectory free to leave and re-enter the feasible region.

Frozen process and controller provenance. The executor and the reference law are the same frozen artifacts used throughout: R_θ is the GuacaMol-trained checkpoint of Section E.2, not retrained or fine-tuned for this benchmark. Only the task-specific controller differs, and the docking-facing controller was frozen before the reported panel was launched. Feasibility predicates (QED, SA, similarity, chemical validity) are evaluated before the protein oracle, so docking calls are spent only on molecules already satisfying the benchmark constraints.

Oracle budget and reported score. Each reported COMPOSE entry consumes 250 docking-oracle evaluations for its cell, and the score reported for that cell is the best feasible docking score obtained within that budget. The published GenMol and InVirtuoGen entries use $1{,}000$ evaluations per run, so COMPOSE uses one quarter of the per-run oracle budget. Budget is counted as docking calls; inexpensive property evaluations are not counted for either method, matching the comparator's accounting.

Feasibility and comparison. A cell counts as feasible when at least one returned molecule satisfies all three endpoint constraints. We report feasibility coverage separately from docking score. Cell-level comparisons include only methods with a reported feasible score for that cell. missing

Docking replicate noise. Repeat evaluation of the same molecule through our QuickVina2 pipeline gives a median replicate variation of 0.70 kcal/mol. Small single-cell differences should therefore not be interpreted as resolved docking improvements.

Comparator values. GenMol and InVirtuoGen entries are published values transcribed without re-docking. A dash denotes that the corresponding method did not report a feasible molecule for that cell.

Table 9: **Superstructure ablation by sampling seed.** Each seed aggregates the ten structural prompts. Quality, uniqueness, and validity are percentages. Diversity is a fraction. Both arms were compliant on all 1,000 attempted outputs for every seed.

Seed	Law	Quality	Uniqueness	Diversity	Validity
0	Learned	41.8	98.0	0.724991	100.0
0	Uniform	26.3	91.3	0.711823	100.0
1	Learned	35.7	97.2	0.724446	100.0
1	Uniform	23.5	89.1	0.706222	100.0
2	Learned	39.6	96.8	0.725786	100.0
2	Uniform	26.4	90.1	0.707463	100.0

Method	Trajectories/source	Success ↑	Diversity ↑
GrIDDD	20	45.1%	0.283
COMPOSE	8	**55.8%**	**0.4999**

Table 10: **Similarity-constrained QED editing on ZINC-250k.** A molecule is successful if it reaches QED ≥ 0.9 while maintaining Morgan-fingerprint Tanimoto similarity ≥ 0.4 to its source. COMPOSE uses the same GuacaMol-trained R_θ without retraining. GrIDDD results are reported from its published evaluation panel, so the comparison is not paired across identical sources. Diversity is the mean within-source pairwise Tanimoto distance among distinct non-source candidates.

H Additional results and ablations

This section reports supporting analyses and ablations of the executable molecular process and its controllers. Each analysis uses the comparison set and protocol described in its corresponding subsection and is interpreted separately from the headline benchmark results.

H.1 Future-reachability evaluation

The panel was sealed under preregistration and opened once. Sixty-seven held-out source-target pairs were constructed whose target is reachable under the frozen executable process within the declared budget; two were excluded from the primary analysis under a committed amendment, giving 65 pairs, and the full 67 are retained as a sensitivity analysis. All arms share the reference law, the legal edit sets, the target molecule and the edit budget. A myopic policy selects the locally preferred edit; verified lookahead evaluates candidate successors by whether the target remains recoverable within the remaining budget; three intermediate prioritizers restrict which successors are expanded before lookahead is applied. Continuation work is reported as a fraction of the candidates evaluated by all-candidate lookahead, so recovery and cost are read together. The unit is the source-target pair and intervals are paired bootstrap.

Of the 65 pairs, 14 are recovered by all-candidate lookahead and not by greedy, and none by greedy and not by all-candidate lookahead. Because the planner retains the greedy action unless lookahead gives a strict improvement, the comparison is one-sided by construction. We therefore report the paired recovery gain, 21.5 percentage points $[12.3, 33.5]$, and the fraction of greedy's 39 failures that lookahead rescues, 35.9% $[21.2, 52.8]$, rather than a discordance test.

Equal recovery counts do not imply equal rescue sets. The goal-aware prioritizer and all-candidate lookahead both recover 40 targets but not the same 40, overlapping on 13 of 14 rescues; the similarity and reference-probability arms are strictly nested inside all-candidate lookahead's rescue set. Recovery was also resolved by verified path length: the advantage does not widen with path length, and the original hypothesis that it would was tested and not supported. As a sensitivity check, including both pairs excluded under the committed amendment gives $26/67$ for greedy and $41/67$ for all-candidate lookahead, a headroom of 15 rather than 14, with no ordering changes. The panel consumed 1,347 kernel calls.

Oracle	COMPOSE	GenMol	InVirtuoGen	REINVENT	Graph-GA
Albuterol similarity	0.605±0.034	0.561	0.482	0.465	0.407
Amlodipine MPO	0.487±0.035	0.566	0.475	0.452	0.480
Celecoxib rediscovery	0.275±0.040	0.450	0.311	0.334	0.290
Deco hop	0.546±0.011	0.635	0.559	0.552	0.558
DRD2	0.789±0.061	0.976	0.921	0.543	0.751
Fexofenadine MPO	0.682±0.048	0.727	0.641	0.626	0.625
GSK3B	0.422±0.038	0.665	0.451	0.450	0.442
Isomers C7H8N2O2	0.795±0.032	0.478	0.418	0.476	0.681
Isomers C9H10N2O2PF2Cl	0.723±0.016	0.536	0.500	0.541	0.557
JNK3	0.208±0.032	0.485	0.318	0.147	0.171
Median1	0.245±0.019	0.234	0.190	0.190	0.173
Median2	0.154±0.008	0.242	0.160	0.160	0.173
Mestranol similarity	0.426±0.021	0.433	0.337	0.302	0.316
Osimertinib MPO	0.756±0.018	0.811	0.720	0.716	0.718
Perindopril MPO	0.453±0.025	0.512	0.401	0.379	0.410
QED	0.878±0.004	0.924	0.875	0.884	0.887
Ranolazine MPO	0.674±0.029	0.681	0.568	0.497	0.499
Scaffold hop	0.421±0.006	0.503	0.445	0.430	0.434
Sitagliptin MPO	0.236±0.043	0.219	0.248	0.192	0.221
Thiothixene rediscovery	0.237±0.016	0.394	0.278	0.274	0.265
Troglitazone rediscovery	0.253±0.029	0.314	0.219	0.219	0.229
Zaleplon MPO	0.333±0.056	0.456	0.379	0.385	0.375
Mean	0.482	0.537	0.450	0.419	0.439

Table 11: **AUC-Top10 on PMO-1K.** Area under the top-10 score curve across 1,000 oracle evaluations, with higher values indicating better performance. COMPOSE per-objective values are mean ± standard deviation over three reported runs. Baseline values are the published base-generator results of Yang et al. (2026). The mean spans the 22 objectives and excludes valsartan SMARTS.

Arm	Rec.	Rate	Cont./pair	Frac. univ.	Gain ret.
Greedy local decision	26/65	40.0%	0.0	0%	-
R_θ top-1 + lookahead	33/65	50.8%	7.3	34.7%	0.50
Similarity top-1 + lookahead	34/65	52.3%	7.3	34.7%	0.57
Similarity top-2 + lookahead	37/65	56.9%	10.8	52.1%	0.79
Goal-aware h_ϕ top-1 + lookahead	40/65	61.5%	7.2	34.6%	1.00
All-candidate lookahead	40/65	61.5%	20.5	100%	1.00

Table 12: Sealed panel of 65 pairs. *Rec.* denotes targets recovered. *Cont./pair* denotes candidate continuations evaluated per pair. *Frac. univ.* gives this count as a share of the all-candidate lookahead count. *Gain ret.* gives the fraction of all-candidate lookahead's 14 rescues also recovered by the arm.

H.2 MID-TRAJECTORY RETARGETING

We evaluate retargeting on 40 source-disjoint held-out molecules under potency-first and developability-first histories. Each prefix contains three committed edits within a six-edit horizon and is fixed before the eventual joint objective B is constructed, so every post-switch comparison begins from an identical realized prefix with the same remaining budget. Margins are normalized by the held-in interquartile range and the reported objective is the worst margin of the conjunction, with zero denoting satisfaction. At the switch the median worst margin is -1.115 following potency-first prefixes and -1.810 following developability-first prefixes, so neither history already satisfies B.

Retargeting materially redirects both histories, and under the primary verified comparison continuing from the realized state remains preferable to restarting from the source. The mean advantage is $+0.302$ $[0.186, 0.413]$ following potency-first prefixes and $+0.176$ $[0.069, 0.283]$ following developability-first prefixes, with wins on $35/40$ and $29/40$ sources respectively. The clairvoyant controller remains better, giving the negative price-of-surprise effects shown in the table. The effect is smaller than in development, with the primary differences shrinking from $+0.444$ to $+0.302$ and from $+0.290$ to $+0.176$, and the confirmation panel contains genuine source-level losses in both histories, so the

Comparison	History	Mean	Median	95% CI	W/L
Responsiveness	potency	$+0.748$	$+0.794$	$[+0.608, +0.891]$	36/0
	developability	$+1.462$	$+1.357$	$[+1.240, +1.691]$	40/0
Greedy-matched history	potency	$+0.367$	$+0.338$	$[+0.251, +0.482]$	37/3
	developability	$+0.211$	$+0.147$	$[+0.065, +0.361]$	26/14
Value of history (primary)	potency	$+0.302$	$+0.243$	$[+0.186, +0.413]$	35/5
	developability	$+0.176$	$+0.127$	$[+0.069, +0.283]$	29/11
Price of surprise	potency	-0.252	-0.123	$[-0.364, -0.155]$	6/32
	developability	-0.428	-0.291	$[-0.580, -0.288]$	4/36

Table 13: Held-out retargeting panel with 40 source-disjoint sources per history. Rows compare retargeting against continuing the old goal, greedy retargeting against greedy restart from the source, verified continuation from x_3 against verified restart from x_0 (the primary estimand), and verified continuation against a clairvoyant controller given B from the first edit. Entries are paired post-switch differences in the registered objective. Positive values favor the first strategy named. W/L counts sources on which that strategy is better or worse, with ties omitted. Intervals are source-level bootstrap.

benefit of state reuse is empirical rather than guaranteed. Prefix revisiting is negligible, occurring for at most one source per arm and history, and does not explain the effect.

H.3 Pathwise constraints

The pathwise experiment remains developmental: the cLogP corridor was selected following an earlier feasibility screen, and a separately frozen held-out confirmation has therefore been specified but not opened. Endpoint-only control constrains only the returned molecule; pathwise control removes any successor outside the corridor before the transition is sampled; a terminal-penalty arm changes the objective but leaves the transition support unchanged. All arms share the reference law, objective, horizon, sources, seeds and evaluation budget, and the unit is the source molecule with source-clustered intervals.

On 24 eligible sources, endpoint-only trajectories return to the corridor after leaving it in $16/24 = 0.667$ $[0.458, 0.833]$ cases under greedy control and $14/24 = 0.583$ $[0.375, 0.792]$ under verified control; the conditional fractions are $16/24$ and $14/23$, so the conditional denominators remain large enough that the unconditional and conditional prevalence estimates agree closely. These excursions are not confined to the boundary: their median depth is 0.69 cLogP units and their median duration is four of the six committed edits. An earlier protected-motif experiment produced zero such excursions, confirming that the estimand need not be positive simply because intermediate states are observed.

Restricting the support to the corridor remains viable on this panel. The median source retains 79.8% of its legal successors, only one source reaches the prespecified support-tight threshold, no evaluated state has an empty constrained edit set, and one trajectory terminates. Excluding the support-tight source leaves prevalence at 0.652 and 0.565 and terminal costs at -0.121 and -0.041, so the conclusions are unchanged.

The developmental panel does not resolve the terminal-utility cost of imposing the constraint. The paired change in normalized DRD2 utility is -0.105 $[-0.377, 0.099]$ under greedy parity, with 5 wins, 8 losses and 11 ties, and -0.023 $[-0.182, 0.134]$ under verified parity, with 9 wins, 10 losses and 5 ties. Both medians are exactly zero and both intervals span zero, which does not establish equivalence: the greedy interval still permits a loss of 0.377 held-in interquartile-range units. Zero violations under support restriction is an integrity consequence of masking rather than an empirical performance statistic, and the apparent advantage of future-aware control under the mask, $+0.661$ $[0.428, 0.944]$, is a ceiling rather than a null, since greedy pathwise control completed all 24 sources and left no binary headroom.

The held-out confirmation fixes the corridor at $C = [2.3689, 4.4522]$ with the same six-edit horizon and endpoint-only semantics, at $n = 48$ sources. It passes only if both criteria clear at full α under intersection-union: the lower bound of the source-clustered interval on hidden-path prevalence must exceed $1/3$, a threshold fixed before the development run, and the lower bound on the mean paired change in terminal utility must exceed -0.25 held-in interquartile-range units, with -0.20 as a

Constraint handling	Endpoint-feasible runs hiding a path-infeasible intermediate	Paired terminal DRD2 utility
Endpoint filtering (greedy)	$16/24$ $[45.8, 83.3]$	reference
Endpoint filtering (verified)	$14/24$ $[37.5, 79.2]$	reference
Support restriction	0 by construction	-0.023 $[-0.182, 0.134]$

Table 14: Constraint handling on the 24-source developmental panel. The first column counts sources whose returned molecule satisfies the corridor while an intermediate state does not. Support restriction excludes these by construction rather than by measurement. The utility column is the paired change against endpoint filtering under verified parity, with the greedy-parity value -0.105 $[-0.377, 0.099]$. Both intervals span zero, so the panel bounds rather than resolves the terminal-utility cost.

prespecified sensitivity analysis. The panel is sized on the joint criterion, giving power 0.976 at the 0.25 margin and 0.874 at 0.20. No held-out source has been opened.

A value head retrained to be aware of a longer horizon was built, evaluated and rejected on four separate measurements, and is not the head used anywhere in Section G. The consequence for the main text is a claim boundary: any gain from a longer horizon comes from giving the frozen process more edit opportunities, not from a better budget-conditioned value model.

H.4 The medicinal-chemistry validity gate

The gate rejects a returned molecule for a small, explicitly enumerated set of reasons: elements outside {C,N,O,F,S,Cl,Br,I,P}; a halogen carrying hydrogens or more than one heavy neighbour (hypervalent iodine such as `[IH4]`); sulfur carrying hydrogens with two or more heavy neighbours; phosphorus that is neither phosphate nor phosphonate; radicals; formal charge beyond ± 1; a ring larger than seven whose atoms are mostly *not* shared with another ring; a nitrogen-nitrogen bond inside a three- or four-membered ring; and cumulated double bonds.

Two of these conditions were narrowed after calibration refuted a broader version. A flat ring-size cutoff was rejected because the benchmark seeds themselves report 7- and 8-membered SSSR envelope rings over fused tricyclic cores, so ring size alone cannot separate a fused drug scaffold from a macrocycle; the condition now applies only to rings whose atoms are largely unshared. A minimum-ring-size condition for pendant systems was rejected because a published InVirtuoGen winner carries a cyclopropane; that minimum was moved out of the validity gate and into the definition of the specific construction operation it belongs to.

The calibration is a test rather than a claim: the gate must accept all fifteen benchmark seeds and all twenty-five published InVirtuoGen winners, and reject the implausible molecules. The suite also asserts that ordinary sulfur chemistry (thiophene, thioether, sulfone) still passes, which is what prevents the hypervalent-sulfur condition from over-rejecting.

We distinguish two gates. A *pathwise* gate asks only whether a state is an executable molecule and is applied to every intermediate; the full gate, which additionally imposes QED, SA and similarity, is applied only to returned molecules. Conflating them is not conservative: applying the full gate to every intermediate prevents a trajectory from passing *through* a temporarily unattractive molecule, and measurably degraded search.

H.5 Selection on locked fragment-program panels

We compared each deployed finite-panel selector with uniform selection over the same distinct eligible endpoints. The comparison reuses all saved proposal panels from the locked motif, decoration, and linker evaluations. Each task has ten prompts, three generation seeds, and 100 attempted output slots per prompt and seed. An empty eligible panel remains a failed slot, and a singleton panel gives the same result under either selector. Proposal construction, native model-support admission, structural checks, and the evaluator were unchanged. These are retrospective selector comparisons on saved panels, not new molecular-generation campaigns. Scaffold morphing reuses linker outputs and contributes no independent sampling unit.

Task	Selector	Quality	Unique	Diverse	Valid	Fidelity	Faithful yield
Motif	Deployed reference score	42.63	93.63	0.6640	99.97	99.97	45.47
	Uniform	37.20	93.53	0.6756	99.97	99.97	39.43
Decoration	Deployed reference score	36.70	93.67	0.5626	100.00	100.00	42.37
	Uniform	34.47	95.00	0.5585	100.00	100.00	38.40
Linker	Deployed score and novelty	31.53	81.03	0.5636	100.00	100.00	36.90
	Uniform	24.00	73.47	0.5744	100.00	100.00	30.67

Table 15: **Selection on identical locked fragment-program panels.** Quality, uniqueness, validity, prompt fidelity, and faithful quality-pass yield are percentages. Diversity is a fraction. Each row aggregates 3,000 attempted output slots over ten prompts and three generation seeds. Benchmark quality is uniqueness-adjusted. Faithful yield divides prompt-faithful outputs passing the existing per-molecule quality criterion by all attempted slots. The linker contrast changes reference-score weighting and the fourfold preference for previously unseen endpoints together.

After averaging generation seeds within each prompt, the deployed-minus-uniform quality difference was 5.43 percentage points for motif (95% prompt-bootstrap interval 3.17 to 7.87), 2.23 for decoration (0.60 to 4.07), and 7.53 for linker (3.57 to 11.50). The prompt was the structural replication unit. Motif improved on nine prompts and tied on one, decoration improved on five, tied on three, and favored uniform on two, and linker improved on seven and tied on three. Uniform selection increased motif and linker diversity by 0.0115 and 0.0108, respectively, and increased decoration uniqueness by 1.33 percentage points. The linker effect therefore does not isolate the native reference score from the deployed novelty preference.

Prompt	Motif	Decoration	Linker
Baricitinib	+7.33	-0.33	+16.00
Cyclothiazide	+3.33	0.00	0.00
Eliglustat	+12.67	-0.33	+15.00
Erlotinib	+3.00	+3.67	+1.00
Futibatinib	+4.67	+7.67	+13.33
Lesinurad	+1.00	+6.67	+6.33
Liothyronine	+4.67	+2.67	+11.33
Lovastatin	0.00	0.00	0.00
Maribavir	+10.33	+2.33	0.00
Spirapril	+7.33	0.00	+12.33

Table 16: **Prompt-level quality differences for locked panel selection.** Entries are deployed minus uniform quality in percentage points after averaging the three generation seeds for each prompt. Negative and tied results are retained. Scaffold morphing shares linker outputs and is not repeated.

The saved receipts include all eight offers in attempted order, including failures and duplicates. We retrospectively restricted each panel to its first one, two, four, or eight attempted offers, then reapplied the original eligibility and deduplication rules. The one-offer selectors coincided, as required when both see the same sole eligible proposal. At four attempted offers, deployed quality exceeded uniform quality at eight offers in every task (Table 17). This is a retrospective truncation of proposals from the eight-offer generator, not a new smaller-pool generation campaign or a measured wall-clock speedup. Repeated selector draws from these panels are not independent generation seeds.

Task	Offers	Deployed quality	Uniform quality	Deployed validity	Uniform validity
Motif	1	26.70	26.70	75.13	75.13
	2	34.37	33.30	92.80	92.80
	4	40.53	37.30	99.20	99.20
	8	42.63	37.20	99.97	99.97
Decoration	1	31.93	31.93	91.43	91.43
	2	35.13	34.03	99.13	99.13
	4	36.70	34.10	99.93	99.93
	8	36.70	34.47	100.00	100.00
Linker	1	21.83	21.83	92.20	92.20
	2	26.27	23.43	99.20	99.20
	4	29.73	24.07	99.97	99.97
	8	31.53	24.00	100.00	100.00

Table 17: **Retrospective attempted-offer prefix analysis.** Quality and validity are percentages over the original attempted output slots. The offers column counts proposals attempted before eligibility checks and endpoint deduplication. Empty panels are failures. The eight-offer rows reproduce the locked full-panel comparison.

H.6 Structured-proposal ablation in PMO

Design and completed-run comparison. We compared the deployed structured proposer (**A**) with length-matched uniform legal edit chains (**B**) on albuterol similarity, celecoxib rediscovery, GSK3B, `isomers_c9h10n2o2pf2cl`, ranolazine MPO, and scaffold hop. The design specified three optimization seeds per objective and a budget of 1,008 charged oracle calls per campaign. We reused the completed A campaigns without additional oracle calls. Both metrics use each campaign's own `canary_v1.json` over its first 1,000 resolved oracle receipts. No prescreening was used.

Proposal mechanism	Final Top-10 ↑	AUC-Top10 ↑
COMPOSE	**0.621**	**0.503**
Uniform legal edit chains	0.558	0.465

Table 18: **Structured proposals under a matched editing allowance.** Means over the same 14 completed objective-seed pairs, with 2 albuterol, 2 celecoxib, 3 GSK3B, 3 C9 isomer, 1 ranolazine, and 3 scaffold-hop replicates. Each cell has equal weight. Metrics use the first 1,000 resolved oracle receipts from a 1,008-charged-call budget without prescreening. Per-objective standard deviations and paired results are reported in Section H.6.

We included only objective-seed cells for which both arms completed the reporting budget, using the same 14 cells for final Top-10 and AUC-Top10. The matched replicate counts were 2, 2, 3, 3, 1, and 3 in the objective order above. The aggregate weights each completed cell equally, so objectives with more completed replicates receive greater weight. Per-objective means and replicate variation are reported in Table 19, and the 14 completed pairs are reported in Table 21.

Uniform legal-chain intervention. For each parent and proposal-channel draw, B first invoked the unchanged structured synthesis, mutation, or recombination path. Successful construction supplied the realized primitive count K. We then discarded that program and constructed a chain with the same planned editing allowance. At each step, we sampled an executable legal native mark from the Active8 action set and recomputed that set on the resulting molecule. The sampler weighted each rewrite family's tentative marks by their count and rejected marks refused by native admission or the executor. The accepted law is uniform over executable native marks, not distinct canonical molecular successors. Multiple marks that produce the same molecule therefore retain their combined probability.

A chain could terminate before K edits if no executable mark was obtained within the implemented sampling limits. Each step allowed at most 64 executor trials after native-mark sampling. A nonempty executed prefix was retained and replayed through the program executor, and a zero-step chain was rejected. The implementation did not restart until success or fall back to the discarded structured program.

This intervention retained the executor, endpoint eligibility, candidate-pool setting, initialization, and online selection and query-allocation rules. It retained conditioning on successful structured construction and the resulting planned length, but replaced the proposed molecular choices, construction priors, and structural information carried by program reuse together. Each arm fitted its own online models and accumulated its own archive from its observed history. B consumed the arbitration random stream, so matching seed labels did not imply matched molecular trajectories. The comparison evaluates the structured proposer as a whole and does not isolate dependency scheduling.

		Final Top-10		AUC-Top10	
Objective	n	A	B	A	B
Albuterol	2	0.7939 ± 0.0087	0.5875 ± 0.1009	0.5689 ± 0.0648	0.4824 ± 0.0329
Celecoxib	2	0.3077 ± 0.0338	0.2542 ± 0.0251	0.2534 ± 0.0207	0.2114 ± 0.0144
GSK3B	3	0.5350 ± 0.0649	0.4322 ± 0.0383	0.4219 ± 0.0382	0.3560 ± 0.0416
Isomers (C9)	3	0.8952 ± 0.0644	0.9144 ± 0.0330	0.7229 ± 0.0164	0.7377 ± 0.0712
Ranolazine	1	0.8108	0.6783	0.6980	0.5672
Scaffold hop	3	0.4613 ± 0.0068	0.4684 ± 0.0093	0.4213 ± 0.0063	0.4250 ± 0.0152

Table 19: **Structured proposals and uniform legal chains by objective.** Values are means $\pm$ sample standard deviations over completed matched optimization seeds. No standard deviation is defined for the single completed ranolazine pair. A is COMPOSE and B is the uniform-chain intervention. Isomers (C9) denotes `isomers_c9h10n2o2pf2cl`.

Results and objective dependence. On the 14 completed pairs, B reduced mean final Top-10 by 0.0630 and AUC-Top10 by 0.0379. COMPOSE had the higher value in 10 of 14 pairs for each metric. Mean differences favored structured proposals on albuterol, celecoxib, GSK3B, and ranolazine, whereas uniform chains had higher means on the isomer objective and scaffold hop. The paired statistics in Table 20 summarize this fixed objective-seed panel. They are exploratory cell-level comparisons, not estimates of generalization over independently sampled objectives.

GSK3B is a learned predictor oracle that was exploitable in every arm. The reported leaders had QED values of 0.03 to 0.23 and SA scores of 6.2 to 6.6, including bare-phosphorus and stacked-hydrazine structures. We retain the objective in the declared benchmark reduction but do not interpret its scores or margins as evidence of chemical quality or biological activity. The AUC calculation integrates trapezoids from $(0, 0)$ and is budget dependent. These 1,000-call values must not be compared with published 10,000-call AUC values.

Contrast	Metric	n	Mean	Median	SD	A higher	p_t	p_W
B−A	Top-10	14	-0.0630	-0.0535	0.0902	10/14	0.0257	0.0245
B−A	AUC-Top10	14	-0.0379	-0.0420	0.0602	10/14	0.0410	0.0494

Table 20: **Paired PMO ablation differences on completed runs.** Differences are intervention minus A. SD describes the finite set of paired differences using divisor n. The two-sided paired t-test and Wilcoxon signed-rank p values are unadjusted. Both metrics use the same 14 completed A/B pairs.

Excluded runs. Four of the 18 planned B runs were excluded from both metrics. One albuterol run failed because B-generated molecules triggered an indexing error in the shared contextual value model, one celecoxib run failed after preemption, and two ranolazine runs did not complete the reporting budget. The albuterol failure was arm dependent. No scores from these runs enter the comparison.

Objective	Seed	Top-10 A	Top-10 B	AUC-Top10 A	AUC-Top10 B
Albuterol	20261925	0.8000	0.6589	0.6147	0.5056
Albuterol	20261983	0.7878	0.5162	0.5231	0.4591
Celecoxib	20260925	0.3316	0.2720	0.2681	0.2216
Celecoxib	20260926	0.2838	0.2365	0.2388	0.2012
GSK3B	20268925	0.4630	0.3945	0.3879	0.3165
GSK3B	20268926	0.5530	0.4310	0.4633	0.3519
GSK3B	20269003	0.5890	0.4710	0.4146	0.3995
Isomers (C9)	20280923	0.8213	0.9484	0.7070	0.8051
Isomers (C9)	20280925	0.9248	0.8825	0.7398	0.6631
Isomers (C9)	20280926	0.9394	0.9121	0.7219	0.7450
Ranolazine	20274926	0.8108	0.6783	0.6980	0.5672
Scaffold hop	20279923	0.4539	0.4593	0.4142	0.4173
Scaffold hop	20279925	0.4626	0.4778	0.4265	0.4425
Scaffold hop	20279926	0.4674	0.4682	0.4231	0.4153

Table 21: **Completed A/B objective-seed pairs.** Both metrics use the same 14 pairs, with 1,000 resolved oracle receipts per run. A is COMPOSE and B uses length-matched uniform legal edit chains.

I BENCHMARK AND BASELINE DETAILS

I.1 QED EDITING BENCHMARK

We evaluate 800 source molecules with initial QED in $[0.7, 0.8]$. A returned molecule succeeds when its QED is at least 0.9 and its radius-two Morgan-fingerprint Tanimoto similarity to the source is at least 0.4. COMPOSE returns eight terminal controlled-trajectory outcomes per source; the published GrIDDD result uses twenty candidates per source. No unsuccessful slot is replaced with a newly sampled candidate. The reported source-level success rate is the fraction of all 800 sources with at least one successful returned molecule.

The COMPOSE evaluation panel and GrIDDD's published panel are not a verified source-matched pair. We therefore report the published GrIDDD result as external context, not as a paired comparison. The different candidate allowances also affect the probability of source-level success. Published values are transcribed without re-derivation.

For COMPOSE diversity, we apply the released Jin evaluation script to distinct successful returned molecules. Its average excludes sources with no successful return and assigns zero to a source with only one distinct success. Among the 446 successful COMPOSE sources, 335 have at least two distinct successful molecules; their mean diversity is 0.4999. Including the 111 one-molecule sources gives 0.376 over all successful sources. The appendix comparison table reports the former conditional estimate. GrIDDD does not specify the evaluator for its headline diversity entry, so the two diversity values are not asserted to be identically computed.

I.2 T4 LEAD-OPTIMIZATION BENCHMARK

The panel from Lee et al. (2025) contains five protein targets, three seed molecules per target, and similarity thresholds $\delta \in \{0.4, 0.6\}$. A returned molecule must satisfy QED ≥ 0.6, SA ≤ 4, and Tanimoto similarity at least δ to the seed. The COMPOSE entry for each cell is its best feasible docking score within 250

The QED, SA, and similarity requirements apply to evaluated endpoints. Intermediate states need only pass the molecular executor. Docking calls define the charged oracle budget; inexpensive endpoint eligibility calculations are recorded separately and do not consume docking calls.

Dashes in the published table mark cells for which no feasible molecule is reported. We carry them through as missing, report feasibility coverage separately from docking score, and do not impute a docking value for an empty cell. GenMol and InVirtuoGen values are transcribed from their published tables; we do not re-evaluate their molecules through the COMPOSE docking pipeline.

I.3 Published record on the QED editing task

The comparison in Table 10 does not exhaust the published record for this task. Stronger values have been reported by graph-to-graph translation methods, listed in Table 22. Their published experiments use the same 800 source molecules, similarity threshold $\delta = 0.4$, and $K = 20$ decoded candidates per source (Jin et al., 2019). The 60.6% entry belongs to VJTNN+GAN; VJTNN itself is reported at 59.9%. Jin et al. (2020) reuse the earlier test set and report 76.9% for HierG2G. These supervised translation methods provide useful task context but have different training information from COMPOSE's goal-independent reference process.

Method	Reported by	K	QED success (↑)	Diversity (↑)
MMPA	Jin et al. (2019)	20	32.9%	0.236
VSeq2Seq	Jin et al. (2019)	20	58.5%	0.331
VJTNN	Jin et al. (2019)	20	59.9%	0.373
VJTNN+GAN	Jin et al. (2019)	20	60.6%	0.376
AtomG2G	Jin et al. (2020)	20	73.6%	0.421
HierG2G	Jin et al. (2020)	20	76.9%	0.477

Table 22: **Further published results on the same similarity-constrained QED editing task.** All rows use the 800-source cohort of Jin et al. (2019) with $\delta = 0.4$ and $K = 20$. These are supervised graph-to-graph translation models trained on parallel source-target pairs curated for the target property, whereas the COMPOSE reference law is trained once without reference to QED and the property enters only through control. The rows therefore measure different problem settings and we do not present them as an ablation of one another, nor is any COMPOSE claim stated relative to them.

J Software, randomization, and compute

J.1 Software and environment

The core training and sampling images use Python 3.11 with `torch` $2.4.0$, `numpy` $1.26.4$, `scipy` $1.13.1$, and `rdkit` $2024.3.5$. We keep the RDKit version fixed within each reported comparison because canonical molecular identity, ring perception, and sanitization affect both executable support and benchmark scoring.

The result record for each experiment identifies its code revision, configuration, seeds, input identities, and output schema. Large corpora, checkpoints, and per-run traces are stored outside the source repository. Reproduction therefore requires the referenced immutable assets as well as the code revision; a fresh checkout alone is insufficient to recover the reported measurements.

J.2 Randomization and statistics

Run seeds are derived rather than drawn, by hashing a protocol tag together with the source molecule and the replicate index and taking the leading eight bytes of the digest. In paired comparisons the arm name is deliberately excluded from the seed, so arms sharing a source and replicate index consume identical randomness and a difference between them cannot be a seeding artifact. Where an arm-specific seed is unavoidable for later replicates, the first replicate is still constructed to be identical across arms.

Intervals are bootstrap intervals resampled over the independent unit named in each experiment. That unit is the source molecule in most cases and the source-target pair in the sealed lookahead panel. Replicate counts differ by analysis, and each result artifact records its own. Where an interval is exact rather than bootstrapped, we state its construction.

J.3 Compute

Reference-law training used a preemptible A10G GPU with 8 CPU cores and 64 GiB of memory; the learned future-value models were trained on the same GPU class. Training and inference resources are reported separately. We distinguish kernel calls, container time, and charged objective evaluations. The sealed lookahead panel consumed $1{,}347$ kernel calls and roughly $24{,}300$ container-seconds.

PMO and T4 have explicit objective-evaluation budgets; the QED editing benchmark specifies a returned-candidate allowance. None of these quantities is interchangeable.