SUGGEST-REVIEW-EDIT-VERIFY: STRUCTURE-PRESERVING EDIT-SPACE REDUCTION FOR ITERATIVE REFINEMENT IN AGENTIC MOLECULAR GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecule generation, which aims to discover molecules with desired characteristics, has recently advanced through natural-language instructions that specify diverse design objectives. Recent LLM-based chemistry agents make generation actionable by leveraging these instructions for iterative generation and refinement. Nonetheless, existing methods struggle to identify effective modifications, limiting their efficacy for jointly satisfying structural requirements and multiple property constraints. To address this challenge, we introduce **SREV-Mol** (*Suggest–Review–Edit–Verify* for Molecular Generation), a structure-preserving framework for iterative molecular refinement. Our approach first translates the natural-language structural requirement into SMARTS (SMILES arbitrary target specification) and generates a verified seed molecule, then iteratively selects effective edits from a reduced space of structure-preserving candidates to refine the molecule. To further improve edit selection, we present Structured Edit Reasoning (SER) and candidate-specific Auxiliary Feature Tokens (AFTs), which provide complementary signals for comparing candidate modifications. Thereby, SREV-Mol enables effective multi-property optimization under structural requirements and up to 14 simultaneous property constraints, substantially reducing broad trial-and-error search over the molecule edit space. Extensive experiments on in-distribution (ID) and scaffold out-of-distribution (OOD) settings show that ours outperforms existing chemistry agents and strong frontier LLMs, despite using an 8B backbone, while continuing to improve with larger refinement budgets.

1 INTRODUCTION

Molecule generation is a fundamental task in computational molecule design for drug and materials discovery, aiming to discover molecules with desired characteristics (Fromer & Coley, 2023; D. Segall, 2012). Among various generation paradigms, one promising direction is to specify molecule design objectives in natural language. A single natural-language instruction can express heterogeneous objectives, encompassing structural requirements and explicit property constraints. Hence, text-to-molecule generation enables molecular structures to be generated while simultaneously accounting for diverse design intents specified in the instruction.

Recent advances in LLMs and tool-using agents have made natural-language molecular design actionable through iterative generation and refinement (M. Bran et al., 2024; Kim et al., 2025; Li et al., 2026c;b; Yang et al., 2026). For example, MT-Mol (Kim et al., 2025) iteratively refines candidates with tool-grounded feedback, but evaluates their effects only after generation, limiting comparison of alternative modifications before edit selection. MolReAct (Li et al., 2026b) provides more structured guidance through reaction-template-based actions, but does not explicitly compare their multi-property effects or preservation of the required structure. Such guidance remains insufficient because edit effects depend on the current molecular structure, and improving one property may degrade others or violate the required structure. Thereby, existing agents often need to broadly search over a large candidate modifications space. This burden is particularly severe in our setting, which combines natural-language structural requirements with up to 14 simultaneous property constraints, whereas existing agents typically consider only a few property objectives.

To address this challenge, this study introduces **SREV-Mol** (*Suggest–Review–Edit–Verify* for Molecular Generation), a tool-based framework that replaces unrestricted molecular search with structure-preserving refinement from a verified seed. At each refinement process, *Suggest* retrieves structure-preserving edits using property-change statistics from matched molecular pair (MMP)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

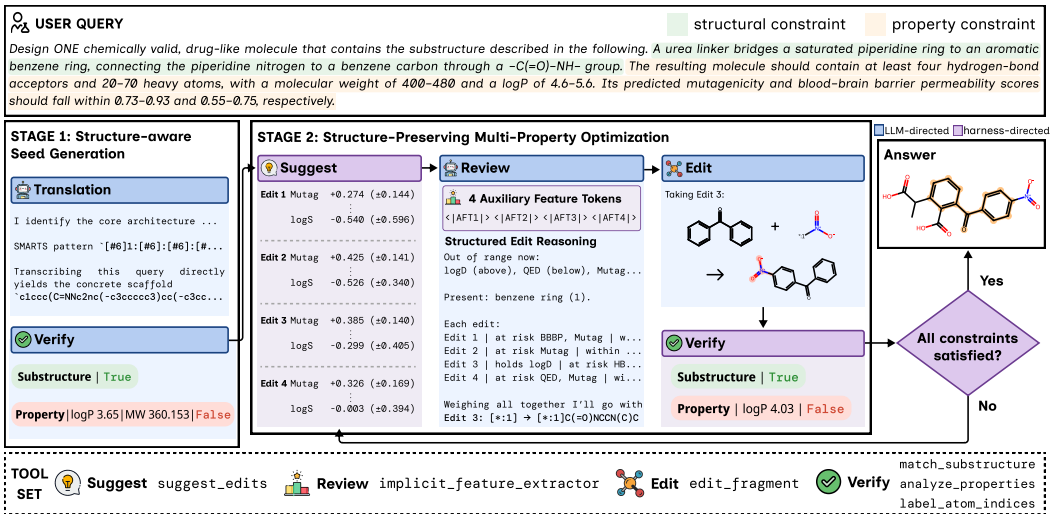

Figure 1: **Overview of SREV-Mol.** Stage 1 produces a verified seed molecule. Stage 2 repeats SREV: *Suggest* filters candidates, *Review* selects an edit with SER and AFTs, and *Edit-Verify* applies and checks it.

analysis (Dalke et al., 2018; Yang et al., 2023; Yi et al., 2024), and *Review* selects among them. *Edit* applies the selected modification, while *Verify* checks the updated molecule against the structural and property constraints. To improve edit selection, SREV-Mol strengthens *Review* with Structured Edit Reasoning (SER), which compares candidates' predicted property ranges and structural changes, and candidate-specific Auxiliary Feature Tokens (AFTs), which provide complementary information for candidate comparison.

As illustrated in Figure 1, SREV-Mol first secures structural validity and then optimizes multiple properties while preserving the verified structure. Stage 1 translates the natural-language structural requirement into a SMARTS (Daylight Chemical Information Systems, Inc.) query and produces a tool-verified seed molecule. In Stage 2, MMP-guided *Suggest* constructs a compact, structure-preserving edit space and ranks candidates by their residual violations across all property constraints. *Review* then uses target-aware SER to compare candidates' predicted property ranges with the target intervals together with their functional-group and atom-count changes. Search-supervised, candidate-specific AFTs learned from molecular search outcomes further complement SER with auxiliary molecular and edit context for candidate comparison.

Experiments on both in-distribution (ID) and scaffold out-of-distribution (OOD) test sets show that SREV substantially improves the molecule-generation capabilities of existing LLMs for jointly satisfying scaffold-level substructure and multi-property constraints, while SREV-Mol further enables more accurate molecular generation through improved structural generation and more effective edit decisions. Additional analyses show that the enhanced *Review* of SREV-Mol enables better edit selection and that its performance continues to improve with additional refinement steps, demonstrating test-time scaling behavior. A detailed comparison with molecular optimization, MMP editing, and LLM-based chemistry agents is provided in Section A.

In summary, the main contributions are:

- **Suggest–Review–Edit–Verify Molecule Generation Framework**. SREV combines structure-aware seed generation with structure-preserving iterative refinement for natural-language scaffold-level requirements and up to 14 simultaneous property constraints.
- **Edit Search-Space Reduction**. *Suggest* step efficiently reduces the large edit space to a compact set of structure-preserving candidates prioritized by MMP-based property changes.
- **Structured *Review* for Edit Selection**. The *Review* step uses SER to explicitly compare candidates' predicted property ranges and structural changes, with candidate-specific AFTs providing complementary information.
- **Empirical Validation of Effectiveness and Generalization**. ID/OOD evaluations show that SREV-Mol improves constraint satisfaction and edit selection, while test-time scaling further improves its refinement performance.

2 PROBLEM FORMULATION

The goal of this paper is to enable joint structural and multi-property molecular design. Formally, given a natural-language structural requirement x^{str} and a set of property target intervals $\{\mathcal{I}_k\}_{k\in\mathcal{K}}$, where $\mathcal{K}$ denotes the set of constrained molecular properties, the task is to find a molecule M that satisfies both the structural and property constraints. Let S denote the reference SMARTS query corresponding to x^{str}, and for each $k \in \mathcal{K}$, let $v_k(M)$ denote the value of property k for molecule M, with target interval $\mathcal{I}_k = [l_k, h_k]$. Either endpoint of $\mathcal{I}_k$ may be unbounded for a one-sided constraint, while $l_k = h_k = t_k$ represents an exact target t_k. The generation objective is

$$\mathrm{Match}(M, S) = 1, \qquad v_k(M) \in \mathcal{I}_k \quad \forall k \in \mathcal{K}, \tag{1}$$

where $\mathrm{Match}(M, S)$ indicates whether M matches the reference SMARTS query S. At inference, the agent receives only x^{str} and $\{\mathcal{I}_k\}_{k\in\mathcal{K}}$, but not S. It therefore infers $\hat{S}$ from x^{str} for structural verification during generation, while final evaluation uses S to prevent an incorrect structural interpretation from being counted as successful. The remaining property violations are quantified as

$$d_k(M) = \frac{\max\{0,\ l_k - v_k(M),\ v_k(M) - h_k\}}{\eta_k}, \qquad D_{\mathrm{prop}}(M) = \sum_{k\in\mathcal{K}} d_k(M), \tag{2}$$

where $\eta_k > 0$ normalizes differences in numerical scale across molecular properties.

3 SREV-MOL: SUGGEST–REVIEW–EDIT–VERIFY FOR MOLECULAR GENERATION

3.1 OVERVIEW AND INFERENCE PROCEDURE

Our SREV framework aims to generate molecules that simultaneously satisfy structural requirements and multiple property constraints through structure-preserving iterative refinement. As illustrated in Figure 1, SREV follows a two-stage procedure: Stage 1 translates the natural-language structural requirement into a verifiable condition and establishes a structurally valid seed, whereas Stage 2 repeatedly refines this seed through the *Suggest–Review–Edit–Verify* loop while preserving the required structure. Within Stage 2, *Suggest* narrows the large edit space to structure-preserving candidates, and *Review* compares these candidates before selecting the final edit and verifying the resulting refinement. Since similarly ranked candidates may differ in uncertainty and molecular context, SREV-Mol enhances *Review* by presenting Structured Edit Reasoning (SER) and candidate-specific Auxiliary Feature Tokens (AFTs). During inference, the LLM generates reasoning or tool calls in LLM-directed steps, while the harness executes predetermined operations.

Stage 1: Structure-aware seed generation *Translation* converts the natural-language structural requirement into a SMARTS query $\hat{S}$ and generates a seed molecule M_0 represented as SMILES (Weininger, 1988). In *Verify*, molecular tools check whether the seed matches $\hat{S}$ and obtain its initial property values. This establishes a structurally valid starting point before property optimization.

Stage 2: Structure-preserving multi-property optimization Each editing round follows the SREV loop: *Suggest*, *Review*, *Edit*, and *Verify*. The harness first retrieves four structure-preserving edit candidates as described in Section 3.2, and the LLM selects one after *Review*. In SREV-Mol, *Review* uses SER and candidate-specific AFTs. The selected candidate is passed to `edit_fragment`, which performs the specified attachment, deletion, or substitution. The harness then invokes `analyze_properties`, `match_substructure`, and `label_atom_indices` to evaluate the edited molecule and prepare the next round. The loop terminates when all structural and property constraints are satisfied or the editing budget is exhausted. Tool interfaces and edit representations are detailed in Section C.1.

3.2 STRUCTURE-PRESERVING CANDIDATE SUGGESTION

At editing round t, *Suggest* constructs a compact edit space from matched molecular pair (MMP) transformations applicable to the current molecule M_t. From a set of molecular SMILES, MMP extraction decomposes each matched pair into a shared *constant* and the two fragments that differ, denoted *from* and *to*. Observations with the same directed transformation $\tau_i = (f_i^{\mathrm{from}} \to f_i^{\mathrm{to}})$ are grouped even when their constants differ. For property k, the resulting changes form

$$\mathcal{D}_{i,k} = \{v_k(M_b) - v_k(M_a) : (M_a, M_b) \mapsto \tau_i\}, \qquad (\mu_{i,k}, \sigma_{i,k}) = \big(\mathrm{mean}(\mathcal{D}_{i,k}), \mathrm{sd}(\mathcal{D}_{i,k})\big), \tag{3}$$

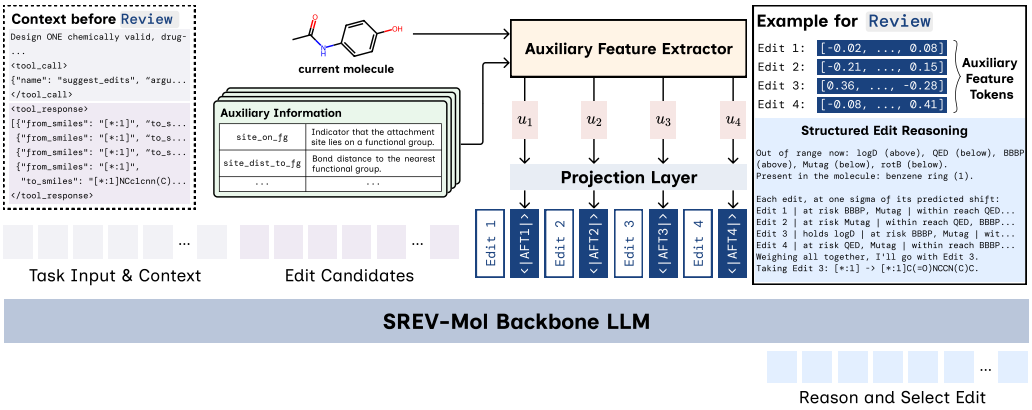

Figure 2: **Overview of the enhanced *Review* step in stage 2.** Auxiliary Feature Tokens derived from $\mathbf{u}_{1:4}$ through a projection layer are inserted after the *Suggest* tokens and provided to the backbone LLM, which generates Structured Edit Reasoning followed by the *Edit*.

so different constants reveal both the typical property effect of the same edit and its variability. The library stores these transformation-level means and standard deviations; construction details are provided in Section B.2. Applying candidate transformation τ_i therefore yields the property estimate

$$\tilde{v}_{t,i,k} = v_k(M_t) + \mu_{i,k}. \tag{4}$$

Candidates are then ranked by their residual violation across all constrained properties:

$$\widetilde{D}_{t,i} = \sum_{k \in \mathcal{K}} \frac{\max\{0,\ l_k - \tilde{v}_{t,i,k},\ \tilde{v}_{t,i,k} - h_k\}}{\eta_k}, \tag{5}$$

where the property-specific scale η_k makes violations across heterogeneous properties comparable.

Candidates must produce valid, connected molecules and preserve the inferred structural query $\hat{S}$. The `suggest_edits` tool returns the top $n = 4$ distinct transformations together with their edit locations, predicted gaps, and MMP property-change statistics, while the associated uncertainty $\sigma_{i,k}$ is retained for the subsequent *Review* rather than used in the initial ranking. Thus, *Suggest* uses empirical MMP statistics to construct a multi-property, structure-preserving edit space rather than directly selecting the final edit. Candidates with similar predicted gaps may still differ in uncertainty and broader molecular context, which are resolved during *Review*. Further scoring details are provided in Section C.2.

3.3 Enhanced Review with SER and AFTs

When the SREV framework is used with an off-the-shelf LLM, *Review* relies on the backbone LLM and the information returned by *Suggest*. SREV-Mol strengthens this step with Structured Edit Reasoning (SER) and candidate-specific Auxiliary Feature Tokens (AFTs). SER explicitly compares predicted property ranges and structural changes, while AFTs provide a context-dependent summary of auxiliary information about the current search state and each candidate edit.

Structured Edit Reasoning. At each round, SER identifies violated properties and their desired directions, summarizes relevant functional groups in the current molecule, and compares the expected effects of candidate edits. For each property, the predicted range $v_k(M_t) + \mu_{i,k} \pm \sigma_{i,k}$ is compared with its target interval. SER distinguishes whether the prediction holds, misses, or only partially overlaps the target, while also considering functional-group and atom-count changes before selecting an edit. The complete label definitions and deterministic target-construction rules are given in Section C.3.

Candidate-Aligned Auxiliary Feature Tokens. The MMP statistics used by *Suggest* average the property changes of a fragment transformation across many molecular contexts. They estimate how the transformation behaves on average, but do not describe the local environment in which it is applied to the current molecule. As illustrated in Figure 2, the auxiliary features include whether the attachment site lies on a functional group and its bond distance to the nearest functional group. These are illustrative examples rather than the complete AFE input. The feature pool also includes

the current property values and constraint margins, MMP property-change means and uncertainties, functional-group and fragment changes, attachment-site topology, transformation support, and edit-history statistics.

The Auxiliary Feature Extractor (AFE) does not treat all of these features as equally relevant at every step. It uses the current-molecule representation to weight the global search-state and candidate-specific features, and then compares the four candidates jointly. For example, when candidates have similar predicted property changes, local crowding and transformation support can help distinguish them; when a property is already close to an upper bound, the remaining constraint margin and the risk of overshooting become more informative; and after an inaccurate previous prediction, uncertainty and edit-history features can receive greater emphasis. This context-conditioned aggregation produces one representation $\mathbf{u}_i$ for each proposed edit, as illustrated in Figure 2.

$$\mathbf{u}_{1:n} = \mathrm{AFE}_\phi(\mathbf{c}, \mathbf{g}, \mathbf{e}^{\mathrm{aux}}_{1:n}), \qquad \mathbf{z}^{\mathrm{AFT}}_i = P_\theta(\mathbf{u}_i). \tag{6}$$

Here, $\mathbf{c}$ is the current-molecule representation, $\mathbf{g}$ contains global search-state features, and $\mathbf{e}^{\mathrm{aux}}_{1:n}$ denotes the candidate-specific auxiliary features for the n proposed edits ($n = 4$ in our setting). The AFE with parameters ϕ produces the candidate representations $\mathbf{u}_{1:n}$, and the shared projector P_θ maps each $\mathbf{u}_i$ to its language-model embedding $\mathbf{z}^{\mathrm{AFT}}_i$. Following prior work on conditioning language models with learned continuous vectors (Li & Liang, 2021), a shared projector maps each $\mathbf{u}_i$ into an Auxiliary Feature Token (AFT), which is inserted immediately after its corresponding *Edit* i entry. The LLM therefore receives a candidate-aligned summary of the auxiliary information most relevant to the current context before generating SER, without requiring every numerical feature to be verbalized.

The AFE is pretrained on depth-5 molecular search trees to predict, for each candidate, the fraction of terminal descendants that satisfy all property constraints. This supervision connects attachment-site topology, transformation support, and search history to the observed outcomes of subsequent edits. After pretraining, the AFE is frozen, while the AFT projector and backbone LLM are jointly trained for candidate review and edit selection. The feature inventory, AFE architecture, pretraining objective, and AFT injection procedure are detailed in Sections C.4 and C.5.

3.4 Training

Training consists of trajectory construction, AFE pretraining, and joint fine-tuning of the LLM and AFT projector. The general SREV framework requires no model training; SREV-Mol learns both Stage 1 structure-aware seed generation and the enhanced *Review* step in Stage 2.

Training instance and trajectory generation Training instances are constructed from 2 million SMILES sampled from the training split. For each SMILES, a scaffold SMARTS query S and corresponding seed SMILES are extracted, and Qwen3.6-27B generates a natural-language description of the seed structure. Between 2 and 14 property constraints are randomly sampled from the 14 optimization properties retained after excluding HIA and formal charge. Starting from the seed, the harness runs the *Suggest–Edit–Verify* loop with beam search and retains trajectories whose final molecules satisfy all constraints. The selected edit at each intermediate state is used as the reference edit $a^\star$. Further details are provided in Sections B.2, B.3 and D.1.

Learning candidate representations for AFTs The AFE is pretrained as part of a candidate-value prediction model, separately from the main LLM. Given a molecular search state and its candidate edits, this model uses regression to predict a success value for each candidate—the fraction of terminal leaves in its depth-5 descendant subtree that satisfy all property constraints. This regression supervision encourages the AFE to encode each candidate's context-dependent downstream potential, making the transferred representations useful for comparing edits beyond their immediate one-step property estimates. After pretraining, only the AFE component is extracted, frozen, and transferred to SREV-Mol to construct the AFTs.

Reasoning targets and joint fine-tuning Training targets are prepared separately for seed generation and edit selection. For Stage 1, an LLM synthesizes the *Translation* and seed-verification reasoning from the reference SMARTS and seed SMILES. For Stage 2, deterministic calculations and templates construct an SER target at each state of a successful trajectory, with $a^\star$ as the selected edit. The frozen AFE computes candidate representations $\mathbf{u}_{1:4}$ during training and inference, while the projector parameters θ and Qwen3-8B backbone are jointly trained. Further details appear in Sections D.2 to D.4.

4 EXPERIMENTS

4.1 SETUP

Evaluation Datasets The main comparison uses 900 in-distribution (ID) and 300 scaffold-out-of-distribution (Scaffold-OOD) instances constructed from ChEMBL (Zdrazil et al., 2024), ZINC (Irwin et al., 2020), and MolDesignBench (Jeong et al., 2026) (Table 1). Here, ID denotes the standard evaluation construction without explicit scaffold exclusion, whereas Scaffold-OOD enforces scaffold disjointness from the training instances. The ID set combines MolDesignBench property constraints with scaffold or functional-group requirements and 300 instances newly constructed from the test molecular pool. The Scaffold-OOD set is scaffold-disjoint from the training instances and focuses on refinement involving at least three of logD, logS, BBBP, and Mutag, whose changes under molecular edits have relatively high variability. Dataset construction, split usage, and structural-description validation are summarized in Section B and Figure 5.

Table 1: **Evaluation-set composition.**

Set	Property	Structure	N
ID	MolDesignBench properties	Scaffold	300
	MolDesignBench properties	Functional group	300
	Test-pool properties	Scaffold	300
Scaffold-OOD	ADMET $\geq$ 3 (Narrow window)	Scaffold	100
	ADMET $\geq$ 3 (Medium window)	Scaffold	100
	ADMET $\geq$ 3 (Wide window)	Scaffold	100

Baselines Baselines include closed general-purpose LLMs with tool-use capabilities, such as GPT-5.6-Sol (OpenAI, 2026) and Claude-Opus-5.0 (Anthropic, 2026), as well as strong open-weight general LLMs, including GLM-5.2 (Zeng et al., 2026), Nemotron-3-Ultra (Blakeman et al., 2026), and Qwen3-235B (Yang et al., 2025). Chemistry-specific agents, including ChemCRAFT (Li et al., 2026a), ChemToolAgent (Yu et al., 2025), and DrugAssist (Ye et al., 2025), use their original tool sets, excluding inaccessible tools, while *LLMs standalone* use no tools. *LLMs w/ Toolset* receive the same tools as SREV without its workflow or harness, while *Off-the-shelf LLMs w/ SREV* additionally follow the framework's enforced *Suggest–Review–Edit–Verify* workflow. The unfine-tuned Qwen3-8B base model is included for comparison with SREV-Mol. Results obtained by additionally exposing textual auxiliary features to these LLMs are reported in Section F.2. Model-specific inference settings, tool access, and stopping rules are documented in Section E.1. The complete data and evaluation documentation is organized in Sections B and E.

Metrics Final molecules are evaluated by Succ.↑, Struct.↑, and Prop.↑, measuring joint, structural, and property-constraint satisfaction, respectively. STRatio↑ averages satisfaction of scaffold element, aromaticity, bond, and ring queries, while PTRatio↑ is the fraction of satisfied property constraints. For methods with edit trajectories, Seed.↑ reports whether the initial molecule satisfies all property constraints, Gain↑ measures normalized PTRatio improvement, and # Edits↓ counts molecular edits; full definitions and output-handling conventions are provided in Section E.2.

4.2 MAIN RESULTS

4.2.1 ID EVALUATION

SREV-Mol achieves the highest overall success on the ID evaluation, reaching 67.8% Succ. compared with 59.1% for the strongest baseline, Claude-Opus-5.0 with SREV (Table 2). The improvement is driven primarily by stronger structural satisfaction: SREV-Mol reaches 82.8% Struct. versus 70.8%, while maintaining comparable property satisfaction (80.6% versus 79.9% Prop.). Thus, the gain in joint success comes largely from better preserving the required structure without sacrificing multi-property satisfaction.

The SREV framework also provides a clear benefit beyond tool access. Relative to the same LLMs with the Toolset alone, SREV increases Succ. from 45.9% to 52.2% for GPT-5.6-Sol and from 56.0% to 59.1% for Claude-Opus-5.0, with substantially larger improvements for GLM-5.2 (16.2% to 34.7%) and Nemotron-3-Ultra (8.7% to 25.2%). This larger benefit for open models indicates that *Suggest* compensates for weaker direct SMILES editing by restricting the edit space, while also reducing the number of edits for most models. Chemistry agents remain near zero in joint success, and the unfine-tuned Qwen3-8B remains limited by weak structural generation and instruction following.

Table 2: **Evaluation on ID Dataset.** Succ./Struct./Prop. denote joint, structural, and property constraint satisfaction; STRatio/PTRatio report the average fraction satisfied. Seed./Gain/#Edits are defined only for methods with an explicit edit trajectory. All applicable metrics except #Edits are reported as percentages; N/A = not applicable. **Bold** = best; ↑/↓ = preferred direction.

	Overall	Structure		Property		Trajectory		
Model	**Succ.↑**	**Struct.↑**	**STRatio↑**	**Prop.↑**	**PTRatio↑**	**Seed.↑**	**Gain↑**	**# Edits↓**
Chemistry Agents								
ChemCRAFT	0.0	16.8	17.1	0.9	40.9	0.7	6.0	**0.130**
ChemToolAgent (gpt-4o)	1.2	17.9	19.1	5.2	57.7	1.1	29.4	2.360
DrugAssist (LLM)	0.0	1.9	1.9	0.6	16.7	N/A	N/A	N/A
LLMs standalone								
GPT-5.6-Sol	14.1	65.2	72.6	21.0	81.3	N/A	N/A	N/A
Claude-Opus-5.0	17.3	67.6	75.0	25.9	84.3	N/A	N/A	N/A
GLM-5.2	2.1	40.4	43.2	3.9	53.3	N/A	N/A	N/A
Nemotron-3-Ultra	1.2	29.0	30.6	2.4	44.8	N/A	N/A	N/A
Qwen3-235B	2.4	37.8	39.8	4.6	50.2	N/A	N/A	N/A
Qwen3-8B	0.2	13.8	14.1	1.0	26.8	N/A	N/A	N/A
LLMs w/ Toolset								
GPT-5.6-Sol	45.9	63.8	70.8	71.0	93.9	12.6	71.2	3.909
Claude-Opus-5.0	56.0	69.1	76.6	79.4	**96.9**	16.8	84.1	3.418
GLM-5.2	16.2	46.3	51.2	32.2	82.0	2.1	57.6	4.230
Nemotron-3-Ultra	8.7	33.6	35.8	15.4	63.2	1.4	30.8	2.937
Qwen3-235B	17.6	40.0	43.1	33.1	78.8	0.9	58.1	3.693
Qwen3-8B	5.0	17.3	17.7	6.6	31.0	0.9	-4.4	0.464
Off-the-shelf LLMs w/ SREV								
GPT-5.6-Sol	52.2	64.8	72.3	76.7	95.3	16.8	76.7	2.902
Claude-Opus-5.0	59.1	70.8	78.1	79.9	96.3	**18.6**	82.7	2.547
GLM-5.2	34.7	44.3	48.3	67.3	87.6	2.8	79.2	2.948
Nemotron-3-Ultra	25.2	32.2	34.3	59.9	79.4	2.6	72.0	2.843
Qwen3-235B	25.1	31.9	33.4	59.8	81.0	0.7	71.0	3.061
Qwen3-8B	12.3	17.4	18.1	29.7	43.6	1.0	36.8	1.469
SREV-Mol	**67.8**	**82.8**	**85.2**	**80.6**	96.6	0.1	**90.5**	3.060

4.2.2 Scaffold-OOD Evaluation

SREV-Mol retains the highest joint success under scaffold and property distribution shifts, reaching 29.0% Succ. compared with 24.0% for the strongest overall baseline, Claude-Opus-5.0 with the Toolset (Table 3). The advantage again appears most strongly in structural satisfaction: SREV-Mol achieves 71.3% Struct.↑ versus 57.7%, while also maintaining higher property success (38.0% versus 36.7% Prop.) and substantially larger refinement gain (82.8% versus 63.9% Gain). Thus, although all methods degrade on unseen scaffolds and multi-ADMET refinement, SREV-Mol preserves its advantage in jointly satisfying structural and property constraints without trading one objective for the other.

Tool use continues to improve property satisfaction, while SREV provides particularly large joint-success gains for open models, including GLM-5.2 (2.0% to 8.3%) and Nemotron-3-Ultra (0.7% to 2.7%) relative to the Toolset alone. The high ID Seed. of GPT-5.6-Sol and Claude-Opus-5.0 also drops substantially under Scaffold-OOD evaluation. The near-zero Seed. of SREV-Mol across ID and Scaffold-OOD is by design: it first generates a minimal molecule satisfying the structural requirement, whereas models with higher Seed. often add extra fragments during seed generation. Thus, SREV-Mol achieves its final performance primarily through subsequent refinement rather than property satisfaction already present in the seed.

4.3 Analysis

4.3.1 Candidate Space and Structure Preservation

Figure 3(a) evaluates edit strategies from reference SMARTS and ground-truth seeds. Generic random edits achieve only 0–7% success, whereas random selection within the MMP candidate space reaches 64.7%, demonstrating that a feasible edit space makes the search tractable. The further gains from greedy selection (68.5%) to beam (94.5%) and exhaustive search (96.4%) show that effective candidate selection within this space also remains important.

The structure-preserving filter makes this candidate space compatible with the required structure (Table 4). Removing it slightly improves Prop.↑ from 80.6% to 83.0%, but reduces Struct.↑ from 82.8% to 53.2% and Succ.↑ from 67.8% to 47.6%. Thus, both a feasible edit space and structural filtering are essential for joint constraint satisfaction.

Table 3: **Evaluation on the Scaffold-OOD Dataset.** Succ./Struct./Prop. denote joint, structural, and property constraint satisfaction; STRatio/PTRatio report the average fraction satisfied. Seed./Gain/#Edits apply only to methods with explicit edit trajectories. All applicable metrics except #Edits are percentages; N/A = not applicable. **Bold** = best; ↑/↓ = preferred direction.

	Overall	Structure		Property		Trajectory		
Model	**Succ.↑**	**Struct.↑**	**STRatio↑**	**Prop.↑**	**PTRatio↑**	**Seed.↑**	**Gain↑**	**# Edits↓**
Chemistry Agents								
ChemCRAFT	0.0	1.3	1.7	0.0	16.5	0.0	3.1	0.230
ChemToolAgent (gpt-4o)	0.0	8.0	9.8	1.7	38.3	0.0	21.9	5.010
DrugAssist (LLM)	0.0	0.0	0.5	0.0	8.3	N/A	N/A	N/A
LLMs standalone								
GPT-5.6-Sol	4.7	51.7	58.7	5.7	59.3	N/A	N/A	N/A
Claude-Opus-5.0	5.3	57.0	63.7	9.7	63.0	N/A	N/A	N/A
GLM-5.2	0.7	16.3	20.8	1.0	23.3	N/A	N/A	N/A
Nemotron-3-Ultra	0.0	11.0	14.2	0.0	17.4	N/A	N/A	N/A
Qwen3-235B	0.0	12.3	15.2	1.3	27.3	N/A	N/A	N/A
Qwen3-8B	0.0	0.3	1.4	0.0	5.9	N/A	N/A	N/A
LLMs w/ Toolset								
GPT-5.6-Sol	17.7	50.7	59.6	28.7	75.7	3.7	50.5	5.183
Claude-Opus-5.0	24.0	57.7	66.2	36.7	**84.3**	4.3	63.9	4.903
GLM-5.2	2.0	25.7	33.3	9.0	57.6	0.3	49.4	4.523
Nemotron-3-Ultra	0.7	11.7	14.2	3.7	41.1	0.0	27.8	2.983
Qwen3-235B	2.7	19.0	24.2	9.3	56.1	0.0	47.0	4.057
Qwen3-8B	0.0	2.0	2.7	0.7	10.7	0.0	2.4	**0.263**
Off-the-shelf LLMs w/ SREV								
GPT-5.6-Sol	16.0	48.3	56.1	26.0	77.6	3.3	49.1	4.950
Claude-Opus-5.0	19.7	57.7	66.7	32.7	81.8	**5.0**	58.3	4.683
GLM-5.2	8.3	18.3	23.3	27.7	70.1	1.0	63.4	4.300
Nemotron-3-Ultra	2.7	9.7	11.3	22.7	63.4	0.0	58.1	4.080
Qwen3-235B	2.0	5.7	8.8	19.0	59.4	0.0	53.6	3.900
Qwen3-8B	0.7	1.0	2.3	5.3	21.3	0.7	18.4	1.520
SREV-Mol	**29.0**	**71.3**	**76.2**	**38.0**	81.9	0.0	**82.8**	4.650

Table 4: **Effect of structure-preserving filtering.** Comparison of SREV-Mol on Qwen3-8B with and without structure-preserving filtering in *Suggest*. All metrics except #Edits are percentages.

	Overall	Structure		Property		Trajectory		
Variant	Succ.↑	Struct.↑	STRatio↑	Prop.↑	PTRatio↑	Seed.↑	Gain↑	# Edits↓
SREV-Mol (w/o filtering)	47.6	53.2	55.6	**83.0**	**97.3**	0.1	**92.7**	3.060
SREV-Mol	**67.8**	**82.8**	**85.2**	80.6	96.6	0.1	90.5	3.060

4.3.2 Review and Edit Selection

To more directly assess whether *Review* improves edit selection, a separate step-level experiment is conducted rather than a full end-to-end ablation. Training examples are extracted from the same successful trajectories used for end-to-end SFT and retain the full trajectory context up to each sampled round, but the model generates only the current *Review–Edit* decision. To distinguish these Qwen3-1.7B models from the full SREV-Mol, the SER-only and SER+AFT variants are denoted as SREV-Mol * *(w/o AFT)* and SREV-Mol *, respectively. All variants are trained on 100K rounds and evaluated on 5K rounds, with depth-5 exhaustive search used to identify alternative successful continuations. Reference Acc.↑ measures agreement with the stored beam-search edit, while Oracle Acc.↑ measures agreement with a shortest continuation found within this bounded depth. Their heuristic-hard subsets retain only rounds where the top-ranked suggestion differs from the reference edit or is absent from the bounded-oracle set, respectively.

Figure 3(b) compares *Direct*, *Freeform CoT*, *Freeform CoT (w/ features)*, SREV-Mol * *(w/o AFT)*, and SREV-Mol * over four random seeds. Both *Freeform CoT* variants underperform *Direct*, whereas SREV-Mol * *(w/o AFT)* substantially improves edit-selection accuracy and SREV-Mol * provides a further gain. These step-level results show that *Review* contributes through better candidate selection rather than reasoning alone, without directly establishing its effect on end-to-end success.

4.3.3 Test-Time Scaling

Figure 4 compares SREV-Mol with GLM-5.2 and Qwen3-8B using either the Toolset or SREV as the edit budget increases. For GLM-5.2, enforcing SREV yields substantially larger gains than the

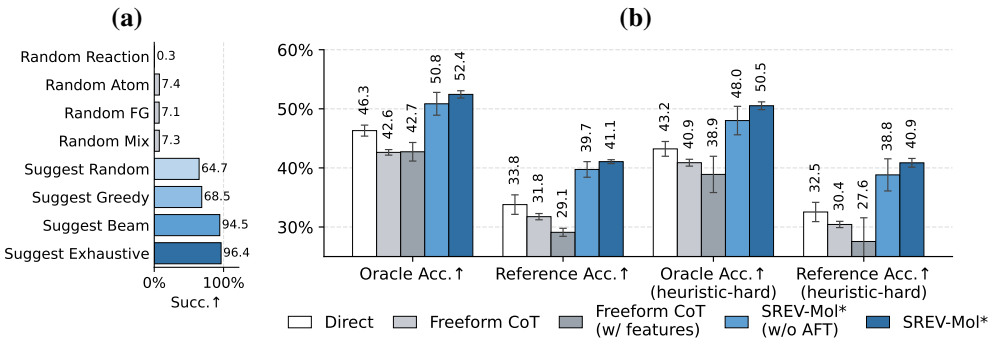

Figure 3: **Ablation on optimization and reasoning strategies.** (a) Multi-property optimization from reference SMARTS and ground-truth seeds on 900 ID instances. All strategies use maximum depth 5 and beam width 4; bars show overall success. FG: functional group; MMP: matched molecular pair. (b) Step-level edit-selection accuracy across reasoning formats.

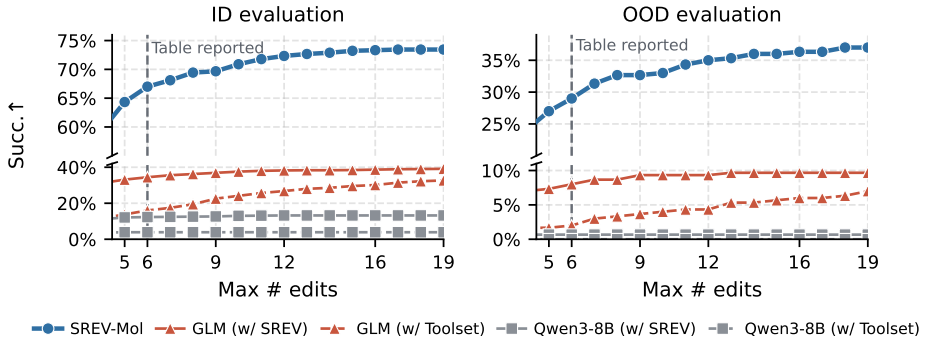

Figure 4: **Test-time scaling with edit budget.** Performance of SREV-Mol, GLM-5.2, and Qwen3-8B on ID (left) and Scaffold-OOD (right) as the maximum number of edits increases.

Toolset alone during the early edits, showing the benefit of structured candidate selection. Moreover, SREV-Mol improves more steeply with additional edits than the general LLMs, indicating that training enables stronger test-time scaling through iterative refinement; the truncation protocol is specified in Section E.4. Protocols for the main comparison, edit-selection evaluation, test-time scaling, and controlled edit-space and *Suggest* studies are provided in Section E; candidate-count and scoring ablations, textual-feature comparisons, AFT analyses, and failure modes are reported in Section F.

5 Limitations

SREV-Mol has several limitations that outline directions for future work. First, constructing the MMP edit library requires substantial upfront property evaluation; a more scalable, transferable approach to building edit statistics would let SREV-Mol extend to a much broader range of properties without this cost. Second, the framework currently supports only a predefined property set and structural constraints expressible as two-dimensional SMARTS; extending it to new properties, three-dimensional geometry, and conformational constraints would broaden its applicability. Finally, since MMP edits do not guarantee synthetic feasibility, incorporating reaction feasibility and synthesis planning into Suggest could yield edits that chemists can directly realize.

6 Conclusion

The SREV framework enables molecular generation under natural-language structural and multi-property constraints through iterative *Suggest–Review–Edit–Verify* refinement. Its MMP-derived edit space filters candidates to preserve the inferred structural query, while SREV-Mol enhances edit selection through SER and candidate-specific AFTs. Strong ID and Scaffold-OOD results, together with ablations, validate the proposed framework. On ID instances, SREV also improves strong closed LLMs without additional fine-tuning, while its Scaffold-OOD gains are more pronounced for open models. These results highlight the effectiveness of combining structural verification, property-guided editing, and iterative refinement.

AI USE STATEMENT

Generative AI was used to construct natural-language description data employed in this work. Specifically, Qwen3.6-27B generated descriptions for training SREV-Mol, `gpt-oss-120b` generated descriptions used to train the round-trip verifier, and DeepSeek-V4-Pro generated descriptions for the test sets. Generated records were automatically checked for required formatting, length limits, and other predefined validity conditions. The test-set descriptions were additionally reviewed manually by the authors. The research task, methodology, and experimental design were developed by the authors. Generative AI was subsequently used to identify potential issues in the proposed design and to assist with searching for related work. It was also used to support implementation of components within an author-designed software framework, draft prompts for synthetic-data generation, and assist with metric-computation and plotting code. The authors debugged and tested the resulting code and inspected intermediate training inputs, including `input_ids` and `position_ids`, to verify that data processing and model training behaved as intended. Generative AI was not used to determine the interpretation of experimental results or the conclusions of the work. Claude-Opus-5.0 and GPT-5.6-Sol were used to draft manuscript text from author-provided outlines and logical structures, improve English readability, translate text, and assist with selected LaTeX tables, figures, and formatting. Generative AI was also used to search for and summarize potentially relevant literature; the authors read the original papers, determined their relevance, and obtained the cited bibliographic records directly. All AI-assisted data, code, claims, citations, figures, and manuscript text were reviewed by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or personally identifiable information, and therefore does not require institutional review board approval. All molecular datasets used in this study are publicly available and were used for research purposes in accordance with their licenses and terms of use. Subject to final license, quality, and safety checks, we plan to release the data artifacts produced and used in this work to support reproducibility. The proposed framework is intended for computational molecular-design research. No generated molecule was synthesized or experimentally validated in this study, and the reported property values and constraint-satisfaction results should not be interpreted as evidence of safety, efficacy, or practical synthesizability. Molecular-generation systems may be misused to propose toxic, regulated, or otherwise hazardous compounds. Any attempt to synthesize or deploy generated molecules therefore requires independent expert review, applicable legal and regulatory compliance, and appropriate experimental safety assessment. Funding and any required competing-interest disclosures will be provided through the appropriate confidential submission channels and disclosed in the final version in accordance with the double-blind review policy.

REPRODUCIBILITY STATEMENT

An anonymous repository is available at `https://github.com/iclr2027-anonymous-13066/mol-sft-data-pipeline`. The dataset sources, splits, construction procedures, and validation process are documented in Section B; implementation and tool details in Section C; training procedures and hyperparameters in Section D; and evaluation settings, metrics, search procedures, and model configurations in Section E. The anonymous repository provides the SFT-data construction pipeline, including trajectory and reasoning generation, together with the ID test-set and Scaffold-OOD data files. Upon acceptance, we plan to release most of the remaining code, processed data, and trained checkpoints, together with dependency specifications or a containerized environment, random seeds, and scripts for reproducing the reported evaluation metrics, tables, and figures.

REFERENCES

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, July 2026. Accessed: 2026-09-14.

Aaron Blakeman, Aaron Thomas, Aastha Jhunjhunwala, Abhibha Gupta, Abhinav Khattar, Adam Rajfer, Adi Renduchintala, Adil Asif, Aditya Vavre, Adriana Flores Miranda, et al. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2606.15007*, 2026.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3): 1096–1108, 2019.

Matthew D. Segall. Multi-parameter optimization: identifying high quality compounds with a balance of properties. *Current pharmaceutical design*, 18(9):1292–1310, 2012.

Andrew Dalke, Jerome Hert, and Christian Kramer. mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. *Journal of chemical information and modeling*, 58 (5):902–910, 2018.

Daylight Chemical Information Systems, Inc. SMARTS – A Language for Describing Molecular Patterns. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html. Accessed: 2026-09-22.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 375–413, 2022.

Jenna C Fromer and Connor W Coley. Computer-aided multi-objective optimization in small molecule discovery. *Patterns*, 4(2), 2023.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35: 21342–21357, 2022.

John J Irwin, Khanh G Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R Wong, Munkhzul Khurelbaatar, Yurii S Moroz, John Mayfield, and Roger A Sayle. ZINC20–a free ultra large-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 60(12):6065, 2020.

Yongjun Jeong, Hanbum Ko, Ye Rin Kim, Chanhui Lee, Rodrigo Hormazabal, Jaewan Lee, Sehui Han, Sungbin Lim, and Sungwoong Kim. Moldesignbench: Evaluating llm-based agent for scenario-grounded molecular design. In *Third Conference on Language Modeling*, 2026.

Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. Mt-mol: Multi agent system with tool-based reasoning for molecular optimization. In *EMNLP (Findings)*, pp. 11544–11573, 2025.

Hao Li, He Cao, Shenyao Peng, Zijing Liu, Bin Feng, Yu Wang, Zhiyuan Yan, Yonghong Tian, Yu Li, and Li Yuan. Agentic reinforcement learning empowers next-generation chemical language models for molecular design and synthesis. *arXiv preprint arXiv:2601.17687*, 2026a.

Tao Li, Kaiyuan Hou, Tuan Vinh, Monika Raj, Zhichun Guo, and Carl Yang. Reinforcement learning with llm-guided action spaces for synthesizable lead optimization. *arXiv preprint arXiv:2604.07669*, 2026b.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp. 4582–4597, 2021.

Yizhan Li, Florence Cloutier, Sifan Wu, Ali Parviz, Boris Knyazev, Yan Zhang, Glen Berseth, and Bang Liu. Mˆ 4olgen: Multi-agent, multi-stage molecular generation under precise multi-property constraints. *arXiv preprint arXiv:2601.10131*, 2026c.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature machine intelligence*, 6(5):525–535, 2024.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, July 2026. Accessed: 2026-09-14.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zaifei Yang, Weiyu Chen, Yaqing Wang, and James Kwok. Probe before you edit: Probing-guided molecular optimization for llm agents in structure-based drug design. *arXiv preprint arXiv:2606.00555*, 2026.

Ziyi Yang, Shaohua Shi, Li Fu, Aiping Lu, Tingjun Hou, and Dongsheng Cao. Matched molecular pair analysis in drug discovery: methods and recent applications. *Journal of Medicinal Chemistry*, 66(7):4361–4377, 2023.

Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1):bbae693, 2025.

Jiacai Yi, Shaohua Shi, Li Fu, Ziyi Yang, Pengfei Nie, Aiping Lu, Chengkun Wu, Yafeng Deng, Changyu Hsieh, Xiangxiang Zeng, et al. Optadmet: a web-based tool for substructure modifications to improve admet properties of lead compounds. *Nature Protocols*, 19(4):1105–1121, 2024.

Botao Yu, Frazier N Baker, Ziru Chen, Garrett Herb, Boyu Gou, Daniel Adu-Ampratwum, Xia Ning, and Huan Sun. Tooling or not tooling? the impact of tools on language agents for chemistry problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7635–7655, 2025.

Barbara Zdrazil, Eloy Felix, Fiona Hunter, Emma J Manners, James Blackshaw, Sybilla Corbett, Marleen De Veij, Harris Ioannidis, David Mendez Lopez, Juan F Mosquera, et al. The ChEMBL database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic acids research*, 52(D1):D1180–D1192, 2024.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Appendix Contents

A Related Work **14**

A.1 Molecule Generation and Optimization 14

A.2 MMP-based Molecular Editing 14

A.3 LLM-based Chemistry Agents 14

B Dataset Construction **14**

B.1 Molecular Corpus and Data Splits 14

B.2 MMP Edit Library Construction 16

B.3 Training Instance Construction 16

B.4 Evaluation Dataset Construction 17

B.5 Structural Description Generation and Validation 17

C Method Details **18**

C.1 Tool Interfaces and Execution 18

C.2 Candidate Scoring 19

C.3 Structured Edit Reasoning Format 19

C.4 Auxiliary Feature Extractor Architecture and Pretraining 21

C.5 Auxiliary Feature Inventory 22

D Training Details **22**

D.1 Trajectory Collection 22

D.2 Reasoning Target Construction 23

D.3 AFT Projection and LLM Fine-Tuning 23

D.4 Training Prompts and Examples 23

E Experiment Details **23**

E.1 Main Comparison Setup 23

E.2 Evaluation Metrics 25

E.3 Edit-Selection Evaluation 25

E.4 Test-Time Scaling Evaluation 26

E.5 Edit-Space and Suggest-Configuration Evaluations 26

F Additional Analyses **27**

F.1 Additional Suggest Results 27

F.2 Textual Auxiliary Features for Off-the-Shelf LLMs 28

F.3 Additional AFT Analyses 28

F.4 Failure Analysis 29

A Related Work

A.1 Molecule Generation and Optimization

Molecular generation and optimization research has largely developed along three directions: text-to-molecule generation, de novo molecule generation, and goal-directed molecule optimization (Edwards et al., 2022; Brown et al., 2019; Gao et al., 2022). Text-to-molecule methods generate molecular representations such as SMILES or SELFIES from descriptions that typically characterize molecular identity or general attributes, rather than explicit structural and numerical property constraints. De novo generation samples molecules from a learned distribution, whereas goal-directed optimization modifies a given molecule toward specified property objectives. These paradigms generally do not jointly support scaffold-level structural requirements and many simultaneous property targets expressed in natural language. Recent constrained-generation methods have begun to combine structural and property conditions, but their structural inputs are often limited to functional groups or explicitly provided machine-readable specifications such as SMARTS. In contrast, SREV-Mol accepts natural-language scaffold-level requirements together with up to 14 simultaneous property constraints, whereas prior agentic methods consider at most four.

A.2 MMP-based Molecular Editing

Matched molecular pair (MMP) analysis quantifies the effect of a local structural transformation by comparing molecule pairs that differ by a single fragment substitution (Dalke et al., 2018; Yang et al., 2023). Such statistics have been widely used in medicinal chemistry to interpret fragment effects and derive transformation rules for property optimization. OptADMET (Yi et al., 2024), for example, uses MMP-derived rules to suggest structural modifications for improving ADMET properties. Prior MMP-based approaches mainly retrieve transformations favorable to individual properties or apply them through fixed rules. In SREV-Mol, *Suggest* instead uses empirical MMP statistics to construct a compact set of edits that preserve the required structure and ranks them by residual violations across all constrained properties. This MMP-derived edit space reduces broad molecular search to a small candidate set, leaving the final context-dependent comparison to *Review*.

A.3 LLM-based Chemistry Agents

LLM-based chemistry agents combine language-model reasoning with external chemical tools to perform molecular design and optimization. ChemCrow (M. Bran et al., 2024) established a general tool-using chemistry agent framework, while recent methods such as MT-Mol (Kim et al., 2025), M^{4}olGen (Li et al., 2026c), MolReAct (Li et al., 2026b), and PROBE (Yang et al., 2026) extend this paradigm to iterative molecular optimization through tool use and feedback. MT-Mol iteratively refines free-form SMILES using tool-grounded feedback, but evaluates a modification only after generation, limiting comparison of alternative multi-property effects before editing (Kim et al., 2025). MolReAct provides a more structured, synthesizable action space through validated reaction templates, but does not explicitly compare how alternative actions affect multiple property constraints or preserve a required substructure (Li et al., 2026b). M^{4}olGen and PROBE similarly guide optimization through BRICS fragments or predefined edit actions, yet existing agents consider at most four simultaneous property objectives and do not jointly support natural-language scaffold requirements. In contrast, SREV-Mol begins from a tool-verified structural seed, uses MMP-derived *Suggest* to construct a structure-preserving multi-property edit space, and performs target-aware candidate comparison through SER. Its search-supervised, candidate-specific AFTs further provide complementary molecular and edit context learned from molecular search outcomes.

A structured comparison of the supported constraints and optimization mechanisms is provided in Table 5.

B Dataset Construction

B.1 Molecular Corpus and Data Splits

The data partition follows the protocol of MolDesignBench (Jeong et al., 2026). Approximately 11M SMILES from ChEMBL (Zdrazil et al., 2024) and ZINC (Irwin et al., 2020) are divided by a Mahalanobis-distance-based split into a 10M-molecule training pool and a 1M-molecule test pool;

Table 5: Comparison of molecular design methods. "Structural requirement" indicates conditioning on an explicit required structure, and "natural-language requirement" indicates that design constraints can be specified in language. #Props. is the maximum number of simultaneous property constraints. "Verification" denotes an explicit post-generation or post-edit check, and "decision output" is the object generated or selected at each decision point.

(a) Supported task and constraints

Method	Model / framework	Task	Structural requirement	Natural-language requirement	# Props.
DrugAssist	SFT LLM (Llama-2-7B)	Multi-property molecule optimization	No	No	2
MT-Mol	Prompted multi-agent LLM (DeepSeek-V3)	Oracle-guided molecule optimization	No	No	4
ChemCrow	Prompted tool agent (GPT-4)	Open-ended molecule generation	No	No	1
OptADMET	MMP rule database	Multi-property molecule optimization	No	No	2
M^4olGen	Multi-agent LLM + RL	Multi-property molecule generation	No	No	3
MolReAct	Frozen LLM + policy head	Synthesizable molecule optimization	No	No	4
PROBE	Prompted multi-agent LLM	Structure-based molecule optimization	Yes	No	3
SREV-Mol (Ours)	SFT agent + AFE (Qwen3-8B)	Language-based structure-constrained optimization	Yes	Yes	14

(b) Optimization mechanism

Method	Edit space	Edit reasoning	Explicit verification	Training	Decision output
DrugAssist	Free-form SMILES	None	No	SFT	Full SMILES
MT-Mol	Free-form SMILES	None	Yes	Prompting	Full SMILES
ChemCrow	Reaction templates	None	No	Prompting	Tool call
OptADMET	MMP transformations	None	No	None	MMP transformation
M^4olGen	BRICS fragments	Naive synthesis	Yes	SFT + GRPO	Fragment edit
MolReAct	Reaction templates	None	Yes	GRPO	Template + building block
PROBE	BRICS fragments	None	Yes	Prompting	Edit site + action
SREV-Mol (Ours)	MMP transformations	SER + AFTs	Yes	AFE pretraining + SFT	SMARTS, seed, and edit rule

further details of this splitting procedure are provided by MolDesignBench. The training pool is used to construct the MMP edit library, the main SREV-Mol training instances and trajectories, the round-trip-verifier dataset, the edit-selection-state dataset used for reasoning-format comparisons, and the dataset used to pretrain the Auxiliary Feature Extractor. The evaluation sets use MolDesignBench instances and molecules from the 1M-molecule test pool as described below. No additional scaffold split is imposed on the ID set; scaffold disjointness from the training instances is enforced only for the Scaffold-OOD set. Figure 5 summarizes these splits and their roles in training, evaluation, and analysis.

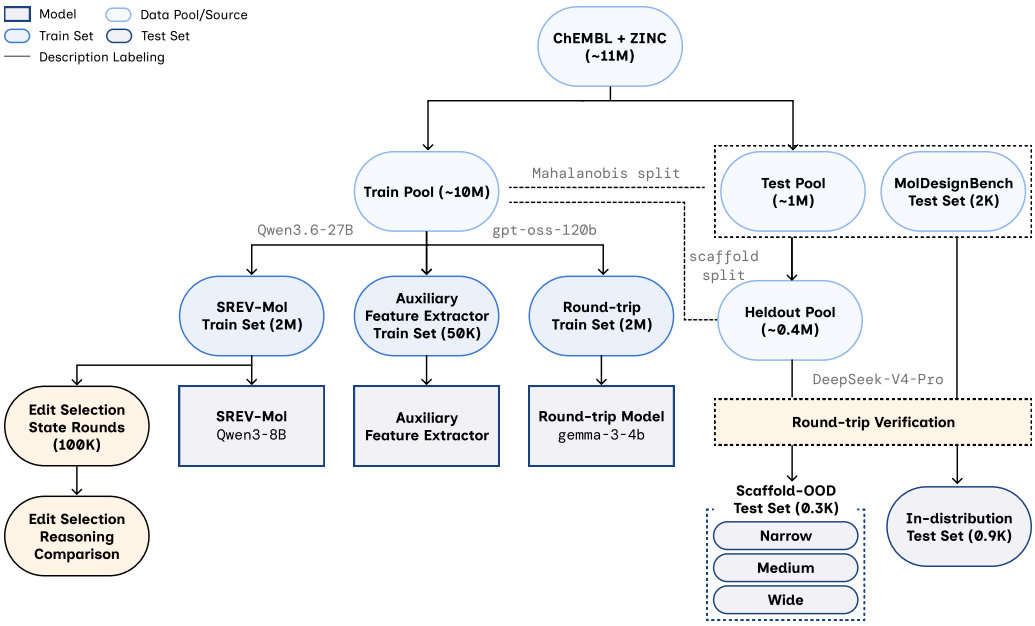

Figure 5: Overview of the molecular-data splits and their use in model training, evaluation, and analysis.

B.2 MMP Edit Library Construction

The edit operations available to SREV-Mol are mined from molecular data rather than manually defined. Approximately one million molecules are randomly sampled from a training pool of 10 million molecules compiled from ZINC and ChEMBL, annotated with 11 physicochemical and 5 ADMET properties. Following stereochemistry removal, structural canonicalization, and deduplication, matched molecular pairs are extracted using `mmpdb`. Each pair shares a common molecular core and differs by a fragment substitution involving one, two, or three attachment points. As illustrated in Figure 6, each pair is represented by its shared constant and a directed *from→to* fragment transformation. Pairs with the same transformation are aggregated across their different constants. For every property k, we compute $\Delta_k = v_k(M_b) - v_k(M_a)$ for all supporting pairs and store its mean $\mu_{\tau,k}$ and standard deviation $\sigma_{\tau,k}$ together with the transformation support N_τ. We retain transformations with $N_\tau \geq N_{\min}$ and set $N_{\min} = 10$ throughout all experiments. These statistics characterize both the typical effect of a transformation and its variability across the molecular constants in which it occurs. Single-attachment fragments are also collected as building blocks. The resulting library contains 139,833 edits and 71,253 building blocks.

B.3 Training Instance Construction

Two million training instances are constructed by uniformly sampling reference molecules with replacement from the training split. For each reference molecule, a Bemis–Murcko scaffold is extracted, retaining rings and their connecting linkers, and represented as a reference SMARTS query S with a corresponding seed SMILES. The `Qwen3.6-27B` model converts the extracted structure into a natural-language description, as detailed in Section B.5. Property constraints are anchored to the reference molecule to ensure feasibility. Of the 16 annotated properties, HIA and formal charge are excluded because of their near-constant distributions, leaving 14 properties for constraint generation and optimization. The numbers of integer-valued and continuous properties are independently sampled uniformly from 1–5 and 1–9, respectively, yielding 2–14 constraints per instance; properties within each group are selected without replacement. Integer constraints use equally probable exact, one-sided, or two-sided bounds, with integer offsets sampled uniformly from 0 to 2 around the reference value. Continuous bounds are obtained from an empirical quantile window of width $q \sim \mathcal{U}(0.1, 0.5)$, randomly positioned around the reference percentile. Bounds extending beyond

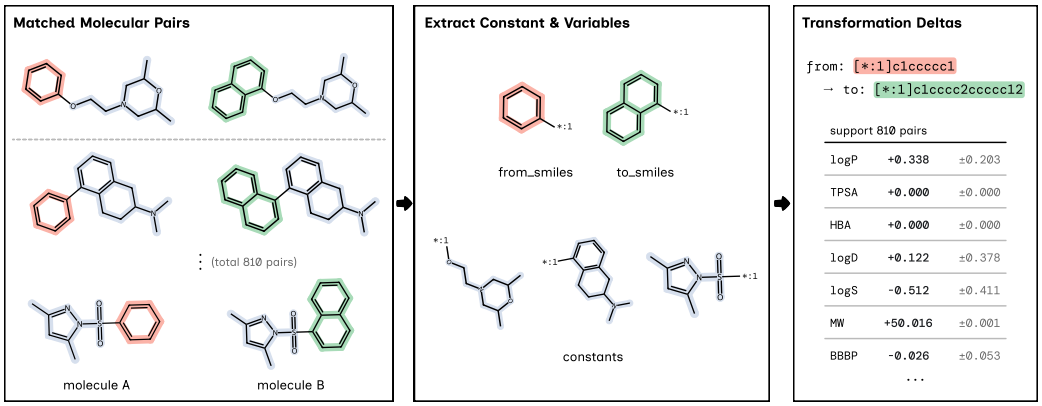

Figure 6: Construction of transformation-level MMP statistics. Matched molecular pairs are decomposed into a shared constant and a directed *from→to* fragment transformation. Property changes from pairs sharing the same transformation but having different constants are aggregated into their mean and standard deviation.

the distribution endpoints are omitted, and numerical rounding preserves inclusion of the reference value. The number and fraction of pool molecules jointly satisfying all property constraints are recorded, with the reference molecule providing a feasible solution by construction.

B.4 Evaluation Dataset Construction

The main end-to-end comparison uses 900 ID instances and 300 Scaffold-OOD instances. ID is used operationally for the standard evaluation construction without explicit scaffold exclusion, while Scaffold-OOD denotes the additional unseen-scaffold condition. Of the ID instances, 600 reuse the property constraints from MolDesignBench: 300 are paired with scaffold-based structural constraints, and 300 with constraints describing one or two functional groups present in the reference molecule. The functional-group descriptions mix direct templates stating the required groups with LLM-generated descriptions. The remaining 300 ID instances are constructed from molecules sampled from the 1M-molecule test pool using the same property-constraint generation procedure as the training instances, without an additional scaffold split.

The Scaffold-OOD set is scaffold-disjoint from the training instances and contains constraints on at least three of four ADMET endpoints: logD, logS, blood–brain barrier penetration (BBBP), and mutagenicity (Mutag). These endpoints are selected because their changes have relatively high standard deviations in the MMP-derived edit statistics, making property-aware reasoning, edit selection, and iterative verification more important than for properties that can often be satisfied during seed generation, such as molecular weight or ring count. For each property, q is the width of the admissible interval in its empirical quantile space; smaller q specifies a narrower constraint on a unit-free scale shared across properties. The set contains 100 instances in each of three window-width strata: $q \in [0.05, 0.1)$, $q \in [0.1, 0.5)$, and $q \in [0.5, 0.7]$.

Scaffold descriptions for both evaluation sets are generated by DeepSeek-V4-Pro, whereas functional-group descriptions use the mixture described above. LLM-generated scaffold descriptions are filtered using the round-trip verification in Section B.5, and all retained evaluation descriptions are manually inspected for consistency with their reference structures.

B.5 Structural Description Generation and Validation

Description generation and verification objective Separate description generators are used for the main training data, evaluation data, and round-trip-verifier training data to reduce correlated generation artifacts. Descriptions for SREV-Mol training are generated by Qwen3.6-27B, scaffold descriptions for the ID and Scaffold-OOD sets by DeepSeek-V4-Pro, and descriptions used to train the round-trip verifier by GPT-OSS-120B. The round-trip verifier takes a natural-language scaffold description as input and predicts the corresponding scaffold SMARTS as output. It is used to audit whether each description faithfully represents its underlying structure. The verifier is used only for

Table 6: Step-level execution structure of SREV-Mol. Stages denote the two high-level phases, whereas the listed steps are operations within them. "Directed by" indicates which component initiates a step; all tool calls are executed by the harness.

Stage	Step	Directed by	Operation	Components
1	*Translation*	LLM	Reasoning	Text
1	*Verify*	LLM	Tool	*Verify* tools
2	*Suggest*	Harness	Tool	*Suggest* tool
2	*Review*	LLM	Reasoning	AFTs and SER
2	*Edit*	LLM	Tool	*Edit* tool
2	*Verify*	Harness	Tool	*Verify* tools

dataset quality control and is not part of SREV-Mol. The verifier is trained to recover the reference SMARTS from its description, and the checkpoint with the lowest loss on the held-out validation set is selected for dataset filtering.

Verifier training data The verifier dataset is constructed from 2,000,000 molecules sampled from the 10M-molecule training pool. These molecules are separated from the data used to train SREV-Mol; the verifier is used only for dataset quality control and does not receive evaluation descriptions during training. Each example contains a system instruction to output only SMARTS, a user turn containing the scaffold description, and an assistant turn containing the corresponding reference `scaffold_smarts`. Only the reference scaffold SMARTS is used as the target; alternative lenient SMARTS candidates are not included. Starting from 2,000,000 description–SMARTS pairs, examples are removed if the target SMARTS does not match the reference SMILES, the SMARTS or description is missing, a description-generation violation is recorded, or the description contains fewer than 10 characters. This filtering retains 1,999,989 pairs. Examples sharing the same scaffold SMARTS are then deduplicated, after which 5,000 examples are uniformly sampled as a held-out validation set, and the remaining 1,992,966 examples are used for training. Validation examples are never used for parameter updates.

Model and training The verifier is initialized from the base model GEMMA-3-4B-PT (Team et al., 2025), a model family different from the description generators and the downstream agent to reduce correlated errors and self-preference. Full-parameter supervised fine-tuning is performed for five epochs. Cross-entropy is applied only to the assistant SMARTS tokens and the end-of-turn token, while the system instruction and user description are masked from the loss, thereby training the verifier to predict the reference SMARTS from the given description. Training uses AdamW with a peak learning rate of 5×10^{-5}, cosine decay, a warm-up ratio of 0.03, zero weight decay, gradient-norm clipping at 1.0, and a global batch size of 96 packed sequences across six GPUs.

Verifier application When applied to the candidate test-set descriptions, approximately 25–35% pass the round-trip verification criterion. Only passing scaffold descriptions are retained, after which the evaluation descriptions are manually reviewed against their reference structures.

C Method Details

C.1 Tool Interfaces and Execution

Structural Representations A task provides a natural-language structural requirement x^{str} and property target intervals, while the reference SMARTS query S is reserved for final evaluation. During end-to-end generation, the agent infers $\hat{S}$ and constructs a seed molecule represented as SMILES. Structural filtering and verification use $\hat{S}$, whereas training-data search and the controlled optimization ablation use the supplied reference query S. These settings separate interpretation errors from the ability to optimize properties while preserving a specified substructure. Table 6 summarizes the two stages, their component steps, and whether each step is directed by the LLM or the harness.

C.1.1 SREV TOOL SET

`suggest_edits`. This tool proposes MMP-derived fragment transformations that move a molecule toward specified property intervals. Given a molecular SMILES and property constraints, candidate edits are screened using the mean property changes aggregated over molecular pairs with the same *from→to* transformation. Candidates are ranked by their predicted remaining distance to the target intervals and checked for applicability and preservation of all supplied structural queries. Each returned candidate specifies the source and replacement fragments, attachment-site indices, predicted gap, and the mean and standard deviation of each constrained property change. These values rank candidates but do not replace structure-specific verification with `analyze_properties`. Figure 7 shows the compact output presented to the agent.

`edit_fragment`. This tool applies fragment attachment, substitution, or removal to a molecular SMILES. Corresponding attachment points in the source and replacement fragments are identified by labelled dummy atoms, while optional anchor indices specify the editing sites and the orientation of transformations involving multiple connections. Products are constructed and validated using RDKit, and the first valid product is returned as a SMILES string.

`analyze_properties`. This tool evaluates a molecular SMILES and returns all 14 properties used for optimization or a requested subset. Ten physicochemical descriptors are calculated using RDKit: exact molecular mass, logP, hydrogen-bond donor and acceptor counts, topological polar surface area, rotatable-bond count, ring count, QED, molar refractivity, and heavy-atom count. Four additional endpoints—logD, logS, blood–brain barrier penetration, and mutagenicity—are predicted using ADMET-AI. These structure-specific evaluations check property constraints independently of the MMP statistics used for edit ranking.

`match_substructure`. This tool checks structural constraints using RDKit substructure matching. The candidate molecule is supplied as SMILES, and queries can be expressed as SMARTS or SMILES fragments. Multiple queries are evaluated independently, allowing their matches to share atoms; the overall result is positive only when every query is present. The tool returns the individual and joint match results used by *Verify*.

`label_atom_indices`. This tool annotates a molecular SMILES with the zero-based atom indices assigned during RDKit parsing. Each index is represented as an atom-map label, establishing the correspondence between atoms in the current molecule and the anchors accepted by `edit_fragment`. Because indices depend on the input representation, labels are obtained from the same SMILES used for editing.

C.2 CANDIDATE SCORING

For candidate edit $a_{t,i}$, the predicted value of property k is $\tilde{v}_{t,i,k} = v_k(M_t) + \mu_{i,k}$. Candidates are scored by the summed normalized target-interval violation

$$\widetilde{D}_{t,i} = \sum_{k \in \mathcal{K}} \frac{\max(0, l_k - \tilde{v}_{t,i,k}, \tilde{v}_{t,i,k} - h_k)}{\eta_k}. \tag{7}$$

An unspecified lower or upper bound is treated as unbounded. The scales η_k are rounded population standard deviations estimated from 40,000 training toolchain records, making violations comparable across properties with different units (Table 7). Ranking uses the property-change means, while the standard deviations are retained for SER. Transformations with positive predicted gap reduction are retained for structural applicability checks. Candidates must produce valid, connected molecules that preserve all supplied structural queries. Up to $K = 4$ candidates are returned, prioritizing distinct source–replacement transformations.

C.3 STRUCTURED EDIT REASONING FORMAT

SER identifies properties outside their target ranges and their required directions of change, together with relevant functional groups in the current molecule. For each candidate and property, the predicted range $v_k(M_t) + \mu_{i,k} \pm \sigma_{i,k}$ is compared with the target interval. The result is `holds` when the entire predicted range lies within the target and `misses` when the two ranges do not overlap.

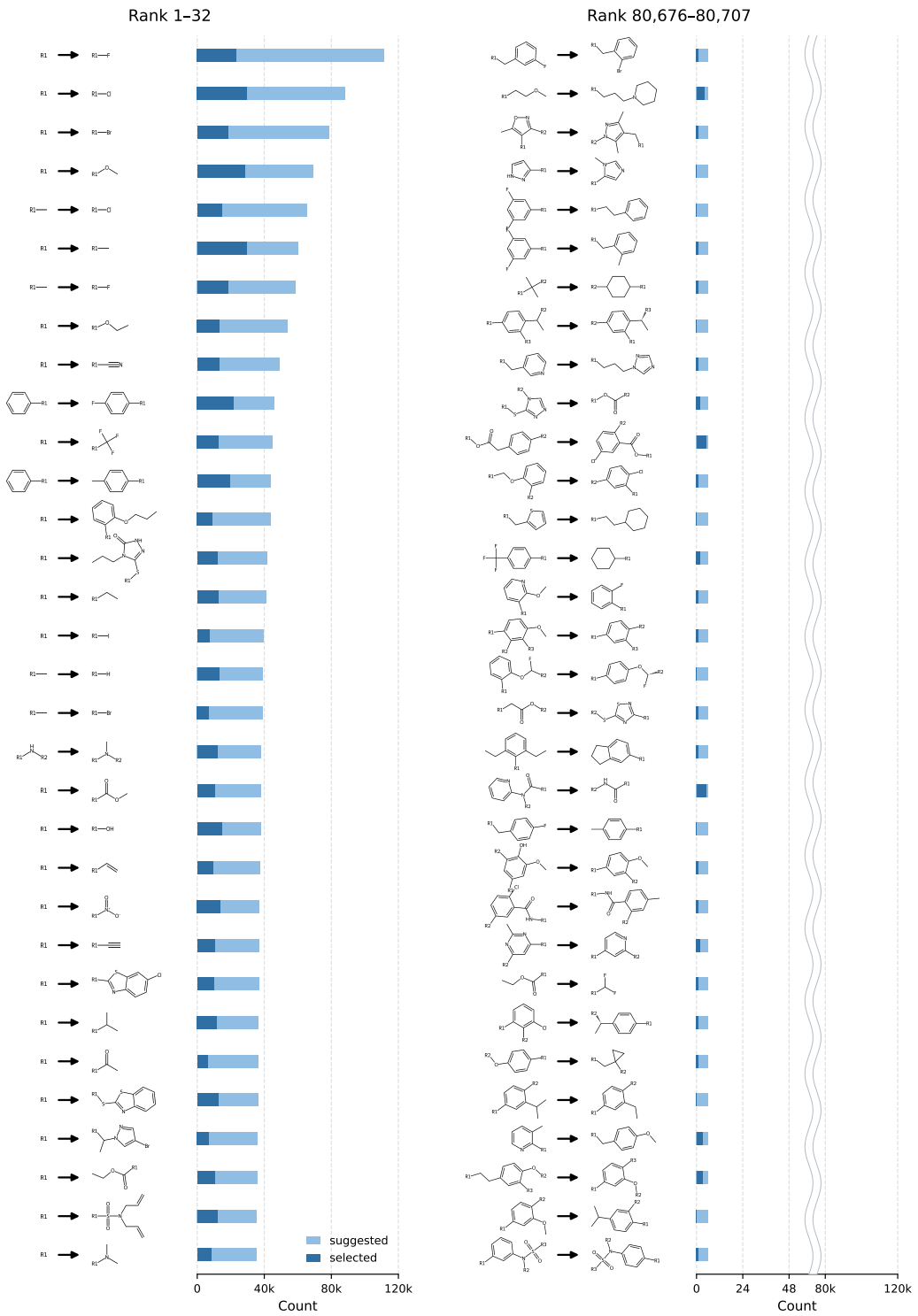

Figure 7: Examples of editing rules obtained through matched molecular pair analysis and the corresponding supporting counts. The *Suggest* utilizes these edits.

When the predicted mean lies within the target, but its uncertainty range crosses a boundary, the result is `at risk`; when the mean lies outside the target but the uncertainty range overlaps it,

Table 7: Fixed property-specific normalization scales used for candidate scoring.

Property	η_k	Property	η_k
MW	86.0	MR	24.0
logP	1.26	Heavy atoms	6.24
HBD	0.85	logD	1.36
HBA	1.62	logS	1.56
TPSA	26.0	BBBP	0.125
Rotatable bonds	1.93	Mutag	0.19
Ring count	1.08	QED	0.15

the result is `within reach`. Changes in functional groups and atom counts are also considered before the candidates are compared, and an edit is selected. During training-data synthesis, these observations are computed using RDKit and deterministic calculations and rendered with predefined templates; the selected edit is taken from a successful reference trajectory. At inference, the trained SREV-Mol LLM generates the review reasoning in the learned SER format. AFT conditioning adds one candidate-specific continuous input token per edit during training and inference; these tokens condition the review rather than construct its text targets. A complete SER example and its candidate-aligned AFT placement are provided in Figure 2; the exact SER labels and comparison rules are summarized here for reproducibility.

C.4 Auxiliary Feature Extractor Architecture and Pretraining

Inputs and feature vectorization. The Auxiliary Feature Extractor (AFE) is pretrained as part of a candidate-value prediction model. Its inputs are a molecular context embedding $\mathbf{c} \in \mathbb{R}^{768}$, a global feature vector $\mathbf{g} \in \mathbb{R}^{88}$ describing the current search state, and an auxiliary feature vector $\mathbf{e}_i^{\text{aux}} \in \mathbb{R}^{167}$ for each of the n candidate edits. The molecular context is obtained by mean-pooling the final hidden states of the frozen MolBERT `molbert_100epochs` encoder over non-padding tokens. Global features describe current property values and bounds, functional-group counts, available sites, search depth, and candidate-set size. Rule features describe property-change statistics, functional-group changes, fragment transformations, and attachment-site context. The complete feature inventory is provided in Table 11.

Each normalized scalar feature x_m is converted into a 128-dimensional token

$$\mathbf{z}_m = \mathbf{b}_m + x_m \mathbf{w}_m, \tag{8}$$

where $\mathbf{b}_m$ identifies the feature and $\mathbf{w}_m$ is its learnable value direction. Thus, the global features form $\mathbf{Z}_G \in \mathbb{R}^{88\times128}$ and the candidate features form $\mathbf{Z}_R \in \mathbb{R}^{n\times167\times128}$. Features are normalized using statistics from the training split, clipped to $[-8, 8]$, and masked when missing, inapplicable, or introduced by candidate padding.

Context-conditioned feature gating. The molecular context acts as a single query over the concatenated global and candidate feature tokens. With query and key dimension 64, the attention weights are

$$\boldsymbol{\alpha} = \mathrm{softmax}\left(\frac{(W_q\mathbf{c})^\top W_k[\mathbf{Z}_G; \mathbf{Z}_R^{\text{flat}}]}{\sqrt{64}} + \mathbf{M}\right), \tag{9}$$

where $\mathbf{M}$ masks invalid features and padded candidates. Each token is weighted by its context-dependent relevance. Pooling the weighted global tokens produces $\mathbf{r}_G \in \mathbb{R}^{128}$, while pooling the 167 weighted auxiliary tokens of candidate i produces an intermediate representation $\mathbf{r}_i \in \mathbb{R}^{128}$. Because the attention is normalized jointly over global and candidate tokens, the global state also influences the candidate representations.

Candidate comparison and value prediction. The intermediate candidate representations are compared through a four-head self-attention layer, yielding the final AFE representations $\mathbf{u}_{1:n}$:

$$\mathbf{u}_{1:n} = \mathrm{LN}(\mathbf{r}_{1:n} + \mathrm{MHA}(\mathbf{r}_{1:n}, \mathbf{r}_{1:n}, \mathbf{r}_{1:n})). \tag{10}$$

Candidate masks prevent padded entries from participating in this comparison. For value prediction, the molecular context is projected to $\tilde{\mathbf{c}} \in \mathbb{R}^{128}$ and concatenated with the pooled global state and

candidate representation:

$$\mathbf{x}_i = [\tilde{\mathbf{c}}; \mathbf{r}_G; \mathbf{u}_i] \in \mathbb{R}^{384}. \tag{11}$$

A shared $384 \to 480 \to 240 \to 1$ MLP with GELU, dropout 0.1, and a sigmoid output predicts the success value $\hat{q}_i$ for each candidate.

Pretraining supervision and optimization. Training states are collected from molecular search trees expanded to depth 5 with up to four candidates per state. The target for candidate i is the fraction of terminal leaves in its descendant subtree that satisfy the property constraints:

$$q_i = \frac{N^{\text{sat}}_{\text{leaf},i}}{N_{\text{leaf},i}}. \tag{12}$$

The value-prediction loss is

$$\mathcal{L}_{\text{value}} = \frac{1}{|\mathcal{C}|} \sum_{(s,i)\in\mathcal{C}} (\hat{q}_{s,i} - q_{s,i})^2, \tag{13}$$

where $\mathcal{C}$ contains valid, unpadded candidates. The dataset contains 2,474,659 decision states from 46,224 search-tree groups, split by originating instance into 2,229,918 training, 120,725 validation, and 124,016 test states. Optimization uses AdamW with learning rate 6×10^{-4}, weight decay 0.01, batch size 1,024 per process, and gradient-norm clipping at 5.0 for 15 epochs; the checkpoint with the lowest validation loss is selected.

Transfer and AFT projection. After pretraining, feature vectorization, context-conditioned gating, candidate pooling, and candidate self-attention form the transferred AFE. The pooled global representation and final value-prediction MLP are not passed to the language model, while the AFE and MolBERT remain frozen. Each final candidate representation $\mathbf{u}_i \in \mathbb{R}^{128}$ is mapped to the 4096-dimensional language-model embedding space by a shared projector:

$$\mathbf{h}_i = W_2\ \text{GELU}(W_1\mathbf{u}_i + \mathbf{b}_1) + \mathbf{b}_2, \qquad 128 \to 4096 \to 4096. \tag{14}$$

The projected vector is RMS-normalized and scaled to the initial token-embedding magnitude ρ:

$$\mathbf{z}_i^{\text{AFT}} = \rho \frac{\mathbf{h}_i}{\max\left(\sqrt{\frac{1}{4096}\sum_{j=1}^{4096} h_{i,j}^2}, 10^{-6}\right)}. \tag{15}$$

The projector is optimized jointly with the language model.

Candidate alignment and embedding injection. An ordered block of `Edit i:` placeholders is appended after each complete `suggest_edits` candidate list. The ordinary embedding at the i-th placeholder is replaced with $\mathbf{z}_i^{\text{AFT}}$, preserving candidate order across sequence packing and truncation. Placeholder positions are excluded from the language-model prediction loss, and no placeholder is added for padded candidates. The same candidate-aligned injection is applied during training and inference before SER generation.

C.5 Auxiliary Feature Inventory

Table 11 lists the global search-state and candidate-level features supplied to the AFE.

D Training Details

D.1 Trajectory Collection

Training the main LLM's edit decisions requires trajectories that reach molecules satisfying both structural and property constraints. Because exhaustive exploration is prohibitively expensive at this scale, beam search with width 4 and at most eight editing steps is run for each instance constructed from the 2 million training SMILES. This collection is distinct from the depth-limited search used to pretrain the AFE (Section C.4). Each search starts from a scaffold seed that contains the required structure, with the reference query S enforced as a hard constraint throughout the search. The harness executes the *Suggest–Edit–Verify* loop without LLM decisions or a *Review* step. For every molecule in the beam, up to four suggested edits are applied; invalid products, products that

do not match S, and duplicate canonical SMILES encountered at the current or previous depths are removed. Surviving products are evaluated using the property tools and ranked by the measured property gap $D_{\text{prop}}(M)$ rather than the predicted gap from *Suggest*. Products with missing required property values are excluded. If no product satisfies all constraints, the four products with the smallest gaps are retained across all beam expansions for the next step. Search terminates upon success, exhaustion of the editing budget, or absence of eligible successors. For each successful search, the stored tool chain contains the initial verification and the successive suggestion lists, selected edits, and post-edit verification checkpoints along a single seed-to-product path. Only trajectories whose final verification satisfies all constraints are retained for main-model training, and the selected edit at each state serves as the reference edit $a^\star$.

D.2 Reasoning Target Construction

Structural-Constraint SMARTS Reasoning and Seed SMILES Generation Synthetic reasoning data for inferring SMARTS representations of natural-language structural constraints and generating seed SMILES is produced using an LLM (Qwen3.6-27B). The LLM is provided with the necessary information, including the substructure description, ground-truth SMARTS, and seed SMILES, where the seed SMILES is constructed using the scaffold of a reference molecule. The detailed generation prompt is provided in Figure 8.

Edit Selection Reasoning for edit selection follows SER as described in Section 3.3. To construct training targets, RDKit and deterministic calculations summarize the current state, projected property effects, and structural changes of each candidate; the selected action is the reference edit $a^\star$ from a successful trajectory. These summaries are rendered using predefined templates, whereas the trained LLM generates SER at inference. For AFT conditioning, the candidate representations $\{\mathbf{u}_i\}_{i=1}^n$ are precomputed using the frozen AFE at every editing round and stored with the training data. An example of candidate-aligned SER and AFT conditioning is shown in Figure 2.

D.3 AFT Projection and LLM Fine-Tuning

The resulting trajectories supervise both structural-constraint interpretation and edit selection through SFT. The AFT projection and input-token placement are specified in Section C.4 and Equation (14). The AFE remains frozen, while the projector is trained jointly with the LLM under the SFT objective, allowing the AFTs to support review and edit selection without requiring their underlying features to be verbalized. At inference, the frozen AFE computes candidate representations, and the trained projector maps them to AFTs in the same input positions before the LLM generates SER and selects an edit. AFT conditioning is therefore used during both training and inference, rather than being an inference-only modification.

SREV-Mol is obtained by full fine-tuning of Qwen3-8B on eight B200 GPUs using bf16 precision and FSDP2. Sequence packing is applied up to 32,768 tokens, and AdamW is used with a learning rate of 5×10^{-5}, weight decay of 0.01, and gradient clipping at 1.0. Standard cross-entropy loss is applied only to tokens generated by the model, while tool outputs and other externally provided content are masked from the loss.

D.4 Training Prompts and Examples

Figure 8 shows the structural-reasoning synthesis prompt, while Figure 9 shows the prompts used to synthesize the free-form edit-selection reasoning baselines.

E Experiment Details

E.1 Main Comparison Setup

The end-to-end LLM evaluation allows at most six edits and seven verification rounds, including the initial verification; verification rounds are counted by calls to `analyze_properties`. Chemistry-specific agents use their original tool sets except for tools requiring paid API access, whereas standalone LLMs receive no tools. Specifically, we exclude `RXN4Chem` for forward-synthesis outcome prediction, `ChemSpace` for reactant-price lookup, and `Tavily` for web search from the chemistry-agent implementations. The *LLMs w/ Toolset* setting provides the same tools used by SREV but does

```
## Task
The assistant is STARTING a molecule-design task from scratch. It (a) reads the described structure and writes the whole SMARTS substructure query in one step, (b) transcribes that SMARTS directly into a concrete starting-scaffold SMILES, and (c) verifies / measures / labels the scaffold in one parallel checkpoint.

## Instructions
- Write 3-5 short first-person sentences. DECISIVE FORWARD reasoning — no scratchpad, no "wait", "let me re-read", "actually", "hold on", "hmm".
- Sentence 1-2: describe the required substructure at a HIGH LEVEL — name the rings and how they connect … Do NOT walk it atom-by-atom.
- Then write the FULL SMARTS from the "Committed SMARTS" block below directly, in ONE step, as the machine-checkable form the match_substructure query will use — character for character, not built up fragment by fragment.
- Then TRANSCRIBE that SMARTS straight into SMILES — read it off and write each atom as a concrete atom (aromatic ``[#6]:[#6]`` → ``c``, ``[#16](=[#8])(=[#8])`` → ``S(=O)(=O)``, ``[#7]`` → ``N``/``n``, etc.). Write the string from the "Starting-scaffold SMILES" block below EXACTLY as given and EXACTLY ONCE … Do NOT mention canonicalization or a "canonical form" — the transcription IS the scaffold.
- The aromatic-hydrogen block below decides whether that read-off is mechanical. A SMARTS ``[#7]`` carries NO hydrogen count, so it matches both ``n`` and ``[nH]``. If the block names any atom, say so in that same transcription sentence … because that H is chemistry you supply, not something the pattern told you.
- The committed SMARTS and SMILES are AUTHORITATIVE — commit to them … But do NOT borrow this prompt's vocabulary: never call the pattern "authoritative", "ground truth", or "the committed SMARTS", and refer to what was asked for as "the query" / "the description", never "the prompt".
- Close by stating the three checks you are about to run together: the SMARTS match, the property measurement, and the atom-index labelling. Word that closing sentence DIFFERENTLY every time …
- Do NOT enumerate specific property-target numbers here, and do NOT include the tool calls.

## User Query
{user_prompt}

## Required Substructure (described in the query)
{substructure_description}

## Committed SMARTS (ground truth)
{substructure_smarts}

## Starting-scaffold SMILES = the SMARTS transcribed to SMILES
{seed_smiles}

## Aromatic hydrogens the pattern does NOT carry (RDKit-computed)
{aromatic_h_note}

## Verified ring names (RDKit-computed)
{ring_locant_note}
```

Figure 8: Training-data synthesis prompt for structural interpretation and seed generation, conditioned on the user request, reference SMARTS, seed SMILES, and RDKit-derived notes.

not enforce the framework's workflow through instructions or a harness. The *Off-the-shelf LLMs w/ SREV* setting provides the same toolset and enforces the *Suggest–Review–Edit–Verify* workflow. In the supplementary textual-auxiliary-feature condition, the same types of auxiliary features used by the AFE are additionally exposed as textual tool outputs. The unfine-tuned Qwen3-8B model is evaluated under these settings as the base-model comparison for SREV-Mol. For GPT and Claude models, reasoning effort is set to `medium` in the standalone setting and `high` when tools are enabled. Qwen3-8B and Qwen3-235B use temperature 0.6 and a maximum generation length of 65,536 tokens. GLM-5.2 and Nemotron-3-Ultra use temperature 0.6, disabled thinking, and a maximum generation length of 8,192 tokens. The instructions are minimally adapted to each model's required prompt and tool-call format while preserving the same task definition, available information, and stopping criteria across models. For SREV-Mol, deterministic content such as predefined tool calls and arguments, repeated constraints, and the SMILES before editing is directly appended to the context rather than regenerated by the model. The iterative design process terminates when all constraints are satisfied or the maximum number of editing steps is reached, after which the final molecule is returned as `<ANSWER>{SMILES}</ANSWER>`.

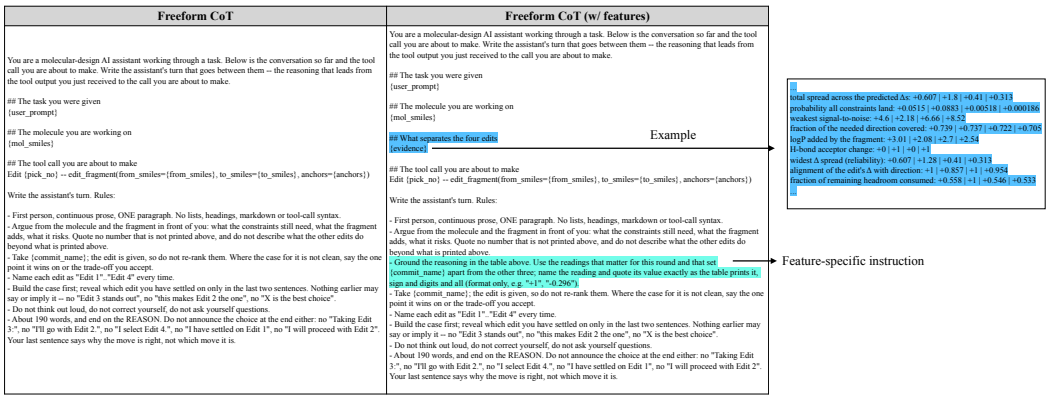

Figure 9: Prompts used to synthesize edit-selection reasoning for the Freeform CoT baseline and its textual-feature variant. Both prompts condition on the task, current molecule, and committed reference edit; the feature-conditioned variant additionally receives candidate-comparison evidence.

E.2 Evaluation Metrics

Constraint Satisfaction Succ.↑ requires satisfaction of the reference structural query S and all property constraints, as specified in Equation (1). Struct.↑ and Prop.↑ measure complete satisfaction of the structural and property constraints, respectively. For STRatio↑, four SMARTS queries separately test element type, aromaticity, bond type, and ring structure; their satisfaction indicators are averaged within each instance and then across the full evaluation set. PTRatio↑ is the fraction of property constraints satisfied within each instance, averaged across the evaluation set. A property constraint is satisfied only when its computed value lies within its specified range.

Trajectory Metrics For methods with an explicit edit trajectory, Seed.↑ indicates whether the initial generated molecule satisfies all property constraints. Let r_{seed} and r_{final} denote the property-satisfaction ratios of the seed and final molecules, respectively. Then Gain↑ is computed as $(r_{\text{final}} - r_{\text{seed}})/(1 - r_{\text{seed}})$, and # Edits↓ is the number of verification rounds minus one. Instances requiring zero edits are excluded when averaging Gain; this also avoids the undefined denominator when the seed already satisfies all property constraints. These metrics are reported as N/A for methods without an explicit edit trajectory. Invalid or missing outputs are counted as failures for Succ., Struct., and Prop., with both STRatio and PTRatio set to zero.

E.3 Edit-Selection Evaluation

The comparison of reasoning formats and AFT conditioning uses dedicated data drawn from the SFT trajectories of SREV-Mol, rather than the main end-to-end evaluation instances. A round is a decision point at which one candidate returned by *Suggest* is selected. Rounds are pooled across all source instances and search depths and randomly split at the round level, rather than by molecule or trajectory. From these pools, 100,000 training rounds and 5,000 held-out test rounds are sampled with a uniform distribution over the selected candidate index in the property-gap ranking within each set. Each example retains the full successful-trajectory context through the sampled round, but the training target contains only the current *Review–Edit* decision. All variants therefore receive matched decision contexts and differ only in the reasoning format and AFT conditioning used for edit selection. *Direct* predicts the edit without an explicit review; *Freeform CoT* uses a free-form review generated from the trajectory context; and *Freeform CoT (w/ features)* additionally receives the feature types supplied to the AFE. The SER-only and SER+AFT variants are denoted SREV-Mol * *(w/o AFT)* and SREV-Mol *, respectively, to distinguish these Qwen3-1.7B step-level models from the full Qwen3-8B SREV-Mol. These sizes refer to individual editing rounds, not complete molecular-design instances, and are distinct from the bounded-search decision states used for AFE pretraining. Each variant is trained for one epoch with four random seeds and evaluated on the same test set; accuracy is reported as mean ± standard deviation across the four runs. Training targets are the reference edits stored in the beam-search trajectories. At each test state, additional exhaustive search to depth 5 identifies edits on shortest successful completion routes. States for which this

search finds no successful depth-5 completion are excluded from the test set. Oracle Accuracy↑ is therefore a depth-5 bounded-oracle metric measuring agreement with an edit on such a route, while Reference Accuracy↑ measures agreement with the stored beam-search commit. When multiple edits tie as shortest successful continuations, all are treated as correct oracle edits.

Heuristic-hard subsets are metric-specific subsets of the 5,000 held-out states. For Oracle Accuracy↑, a state is included if the candidate ranked first by predicted property gap is not an oracle edit. For Reference Accuracy↑, a state is included if that candidate differs from the stored reference commit. The subsets need not contain the same states, because a beam-search reference edit need not belong to a shortest completion route. Each heuristic-hard subset contains 2,145 test rounds.

Interpretation of reasoning-format effects. Freeform CoT's lower accuracy than Direct suggests that free-form reasoning can introduce an additional generation burden: identifying observations that distinguish the selected edit and enumerating their precise values may be harder than selecting the edit itself. SER instead exposes MMP-derived standard deviations as estimates of variability in property changes and summarizes projected constraint satisfaction with `holds`, `misses`, `at risk`, and `within reach` labels. These summaries make uncertainty available for candidate comparison without reconstructing it in free-form prose, although the comparison evaluates SER as a whole and does not isolate the contribution of uncertainty labels. With AFT conditioning, the AFE uses molecular-context-conditioned attention to extract relevant information from the current molecule and auxiliary features without requiring every feature to be verbalized. This learned aggregation complements the explicit SER; the accuracy gains support AFT conditioning but do not by themselves establish this interpretation as its causal mechanism.

E.4 Test-Time Scaling Evaluation

The test-time scaling comparison evaluates SREV-Mol, GLM-5.2 with the Toolset or SREV, and Qwen3-8B with the Toolset or SREV as the maximum edit budget increases to 19. The same ID and Scaffold-OOD evaluation sets and success and property-satisfaction metrics are used as in the main comparison. A single trajectory is generated with a maximum of 19 edits and then truncated one edit at a time to evaluate smaller budgets, rather than rerunning generation separately for every budget. If all constraints are satisfied before 19 edits, generation terminates at that successful state and no longer trajectory is produced.

E.5 Edit-Space and Suggest-Configuration Evaluations

The edit-strategy comparison starts from the reference SMARTS and ground-truth seed for each of the 900 ID instances, isolating multi-property optimization from interpretation and seed-generation errors. All strategies have maximum depth 5, and beam search uses width 4. Random atom attachment randomly samples an attachment site, uniformly samples an atom from {C, N, O, F, S, Cl, Br, I}, and adds the atom through a single bond wherever chemically valid. Random functional-group attachment uniformly samples from the functional groups provided by RDKit, excluding methyl, F, Cl, Br, OH, and NH_2 because they overlap with the atom-level edit set, and attaches the selected group through a valid single bond. The random-mix strategy chooses atom attachment or functional-group attachment with equal probability. Random reaction editing uses the `SynSpace` forward-reaction-template Python library and randomly selects one applicable reaction template at each step. No explicit random seed is set, so the library's default random-number-generator behavior is used. Each non-exhaustive strategy expands four candidates per state. Random search selects among these candidates uniformly, greedy search follows the candidate with the smallest predicted gap, and beam search retains four states ranked by predicted gap; exhaustive search instead expands every applicable candidate. All strategies terminate immediately when a successful molecule is found or when the maximum depth is reached. Products are canonicalized as SMILES, and duplicate states are removed within each search depth. Verification-call counts include evaluation of the initial molecule and all attempted candidates, including invalid and duplicate candidates. Because this controlled comparison starts from the ground-truth structural specification, the structure-preserving filter passes the reference query S to `match_substructure` and removes candidate edits whose products do not contain S. The candidate-count and scoring-method comparisons also use the 900 ID instances with beam width 4 and maximum depth 5. These evaluation searches differ from the width-4, depth-8 beam search used to generate main-training trajectories from 2 million training instances and the depth-5 search used for AFE pretraining. The alternative *Suggest* scores

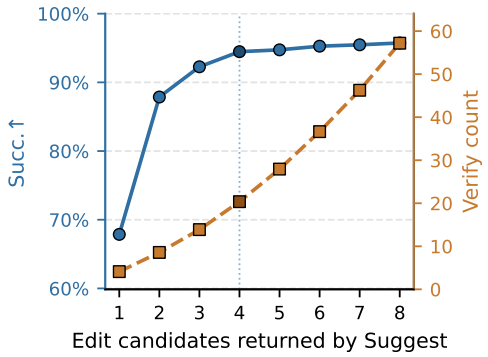

Figure 10: Changes in Succ.↑ and Verify count based on the number of edits returned by *Suggest* in beam search.

Table 8: *Suggest* ranking-score ablation on 900 ID instances using beam width 4 and maximum depth 5. All scores are maximized. For compactness, $g_k(x) = \max(0, l_k - x, x - h_k)/\eta_k$ denotes the normalized distance from value x to the target interval for property k. Higher success and fewer verification calls are preferable.

By strategies	Scoring Method	Description	Calculation Formula	Overall Success (%)↑	Avg. # *Verify*↓
MMPDB beam	count	Counts the number of property constraints satisfied by the expected property values.	$\sum_{k\in\mathcal{K}} \mathbb{1}\left[l_k \le \tilde{v}_{t,i,k} \le h_k\right]$	85.3%	18.53
MMPDB beam	minmax	Sums property gaps normalized by the observed range of each property.	$-\sum_{k\in\mathcal{K}} \frac{\max(0, l_k - \tilde{v}_{t,i,k}, \tilde{v}_{t,i,k} - h_k)}{v_k^{\max} - v_k^{\min}}$	93.6%	20.51
MMPDB beam	worst	Scores an edit according to its largest normalized property violation.	$-\max_{k\in\mathcal{K}} g_k(\tilde{v}_{t,i,k})$	93.0%	21.27
MMPDB beam	prob	Sums the probabilities that the edited properties fall within their target intervals.	$\sum_{k\in\mathcal{K}} \left[\Phi\left(\frac{h_k - \tilde{v}_{t,i,k}}{\sigma_{i,k}}\right) - \Phi\left(\frac{l_k - \tilde{v}_{t,i,k}}{\sigma_{i,k}}\right)\right]$	70.7%	16.83
MMPDB beam	gapstd	Averages the normalized gaps at one standard deviation below and above the expected value.	$-\frac{1}{2}\sum_{k\in\mathcal{K}} [g_k(\tilde{v}_{t,i,k} - \sigma_{i,k}) + g_k(\tilde{v}_{t,i,k} + \sigma_{i,k})]$	93.7%	19.15
MMPDB beam	gap	Sums the normalized gaps between the expected property values and their target intervals.	$-\sum_{k\in\mathcal{K}} g_k(\tilde{v}_{t,i,k})$	94.5%	20.37

and their complete definitions are reported in Table 8; the fixed property-specific scales η_k are listed in Table 7.

F Additional Analyses

F.1 Additional Suggest Results

Number of suggested edits. SREV-Mol returns four candidate edits by default. Figure 10 reports Succ.↑ and the average number of *Verify* calls↓ while varying the candidate count from 1 to 8 during depth-5 beam search on the 900 ID instances. Success increases sharply up to four candidates and improves only marginally thereafter, whereas verification cost continues to grow, making four candidates a favorable trade-off.

***Suggest* scoring method.** *Suggest* ranks candidate edits by minimizing $\widetilde{D}_{t,i}$ by default. Table 8 compares this score with alternative criteria under the same depth-5 beam-search setting. The scores incur similar verification costs, while ranking by $\widetilde{D}_{t,i}$ achieves the highest success rate.

Table 9: ID results for off-the-shelf LLMs using the SREV framework with and without textual auxiliary-feature outputs. These textual outputs are distinct from the learned AFTs used by SREV-Mol. All metrics except #Edits are percentages.

Model	Succ.↑	Struct.↑	STRatio↑	Prop.↑	PTRatio↑	Seed.↑	Gain↑	#Edits↓
Off-the-shelf LLMs w/ SREV								
GPT-5.6-Sol	52.2	64.8	72.3	76.7	95.3	16.8	76.7	2.902
Claude-Opus-5.0	59.1	70.8	78.1	79.9	96.3	18.6	82.7	2.547
GLM-5.2	34.7	44.3	48.3	67.3	87.6	2.8	79.2	2.948
Nemotron-3-Ultra	25.2	32.2	34.3	59.9	79.4	2.6	72.0	2.843
Qwen3-235B	25.1	31.9	33.4	59.8	81.0	0.7	71.0	3.061
Qwen3-8B	12.3	17.4	18.1	29.7	43.6	1.0	36.8	1.469
Off-the-shelf LLMs w/ SREV + Textual Auxiliary Features								
GPT-5.6-Sol	50.7	63.9	71.5	74.4	94.9	15.8	74.3	3.026
Claude-Opus-5.0	58.0	69.9	77.3	80.2	96.9	17.2	83.0	2.597
GLM-5.2	35.9	44.8	48.4	70.3	89.5	2.1	81.7	3.026
Nemotron-3-Ultra	25.7	31.9	33.9	60.9	81.0	1.6	74.2	2.979
Qwen3-235B	24.2	31.8	34.0	60.1	80.8	1.0	70.2	3.001
Qwen3-8B	12.1	17.2	17.6	30.9	45.5	1.4	39.1	1.613

Table 10: Scaffold-OOD results for off-the-shelf LLMs using the SREV framework with and without textual auxiliary-feature outputs. These textual outputs are distinct from the learned AFTs used by SREV-Mol. All metrics except #Edits are percentages.

Model	Succ.↑	Struct.↑	STRatio↑	Prop.↑	PTRatio↑	Seed.↑	Gain↑	#Edits↓
Off-the-shelf LLMs w/ SREV								
GPT-5.6-Sol	16.0	48.3	56.1	26.0	77.6	3.3	49.1	4.950
Claude-Opus-5.0	19.7	57.7	66.7	32.7	81.8	5.0	58.3	4.683
GLM-5.2	8.3	18.3	23.3	27.7	70.1	1.0	63.4	4.300
Nemotron-3-Ultra	2.7	9.7	11.3	22.7	63.4	0.0	58.1	4.080
Qwen3-235B	2.0	5.7	8.8	19.0	59.4	0.0	53.6	3.900
Qwen3-8B	0.7	1.0	2.3	5.3	21.3	0.7	18.4	1.520
Off-the-shelf LLMs w/ SREV + Textual Auxiliary Features								
GPT-5.6-Sol	17.3	48.7	57.1	28.7	77.6	3.7	51.4	4.790
Claude-Opus-5.0	20.3	58.7	67.3	34.3	81.8	4.7	58.5	4.560
GLM-5.2	8.7	18.7	22.6	23.3	67.2	0.3	60.2	4.300
Nemotron-3-Ultra	3.0	7.7	11.5	21.3	61.9	0.0	57.5	4.057
Qwen3-235B	2.0	7.0	10.5	21.3	59.2	0.0	51.6	3.897
Qwen3-8B	0.3	1.3	1.8	6.0	22.1	0.0	20.1	1.423

F.2 Textual Auxiliary Features for Off-the-Shelf LLMs

Tables 9 and 10 compare the standard SREV workflow with a supplementary condition in which off-the-shelf LLMs can query textual versions of the auxiliary feature types used by the AFE. This condition does not use the learned, candidate-specific AFT representations of SREV-Mol. Textual feature access produces only small and inconsistent changes across models and datasets, indicating that simply verbalizing the features is insufficient to reproduce the benefit of integrating learned AFTs into the enhanced *Review* step.

F.3 Additional AFT Analyses

The roles of AFE and AFT are further analyzed from three perspectives: whether AFE focuses on context-relevant features, how the 128-dimensional representation $\mathbf{u}$ is distributed, and whether the backbone LLM uses the candidate-specific representations during edit selection.

For the feature-relevance analysis, round-0 samples from the 5K edit-selection test set are grouped into instances constrained exclusively by ADMET (511), lipophilicity (553), solubility (295), size (162), or polarity (424) properties, resulting in 1,945 instances. Feature importance is measured by masking each feature from the AFE attention computation and recording the decrease in the top-1

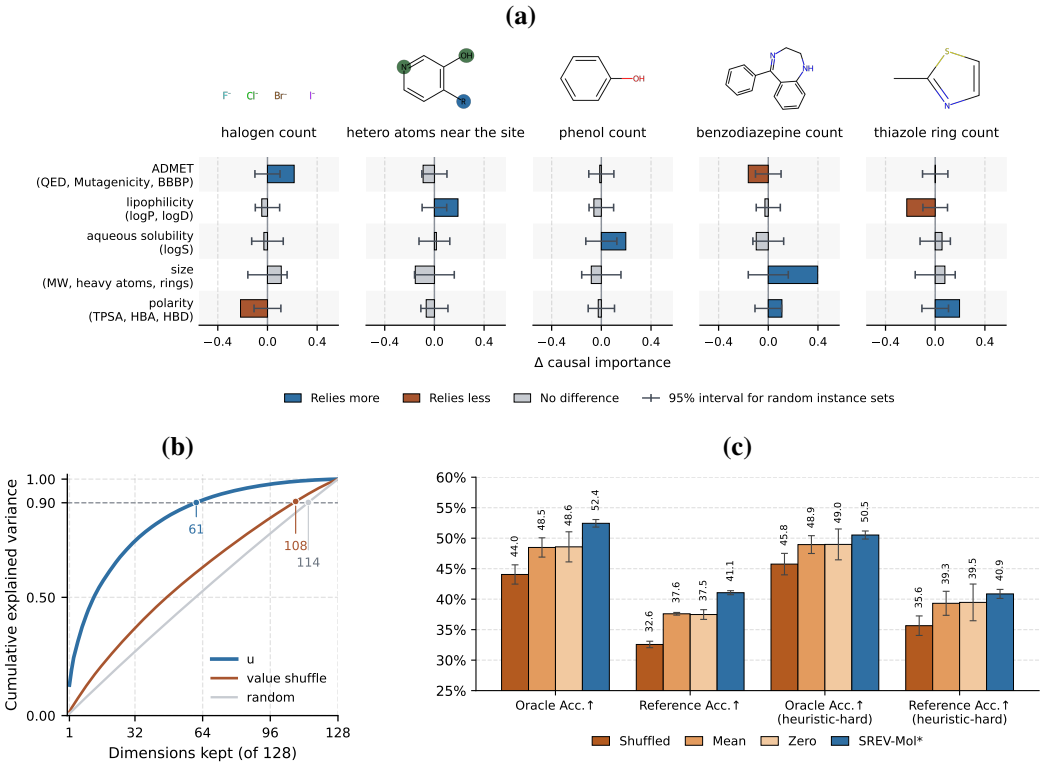

Figure 11: Additional analyses of the AFE representations and AFT conditioning. (a) Feature relevance across property-constraint categories. (b) PCA spectrum of the learned representation and control baselines. (c) Edit-selection accuracy after zeroing, shuffling, or averaging the candidate-specific representations.

versus top-2 predicted success-score margin. A size-matched random subset is sampled from the full pool 4,000 times for each category to construct a null distribution. As shown in Figure 11(a), the resulting relevance patterns align with chemical intuition; for example, benzodiazepine count is associated with size-related constraints, while the thiazole-ring feature is particularly relevant to polarity-related constraints.

Figure 11(b) applies PCA to $\mathbf{u}$ and compares it with value-shuffled and Gaussian-noise baselines. For the original representation, 61 principal components explain more than 90% of the variance. Finally, Figure 11(c) replaces the candidate representations with zeros, shuffles $\mathbf{u}_{1,2,3,4}$ across candidates, or replaces them with their mean. All perturbations reduce edit-selection accuracy, indicating that the AFTs provide candidate-specific information used by the backbone LLM.

F.4 Failure Analysis

We analyze failures of SREV-Mol and Claude-Opus-5.0 by evaluating structural and property constraints independently; the two failure types may co-occur. A structural failure means that the final molecule does not satisfy the reference structural query, whereas a property failure means that at least one requested property interval is not satisfied. Structural complexity is measured from the reference scaffold, rather than the generated molecule. All percentages are failure rates within the corresponding ID or Scaffold-OOD subset.

Structural complexity. For SREV-Mol, structural failure increases from 35.3% (72/204) for bicyclic fused systems to 52.1% (38/73) for tricyclic or larger fused systems on ID instances, and from 30.4% (31/102) to 45.5% (15/33) on Scaffold-OOD instances. Claude-Opus-5.0 remains near 50% across these categories: 50.0% (102/204) versus 50.7% (37/73) on ID and 53.9% (55/102) versus 51.5% (17/33) on Scaffold-OOD instances. Structural failures are associated with larger scaffolds, greater ring sharing, and more stereocenters. The comparisons below are restricted to the 600 ID and

300 Scaffold-OOD instances that specify a reference scaffold, excluding the 300 functional-group instances for which no scaffold is defined; they cover 150 and 86 structural failures of SREV-Mol, respectively. On ID instances, the failure rate of SREV-Mol rises from 12.5% (35/280) to 35.9% (115/320) when the scaffold has at least 20 heavy atoms, from 12.7% (42/330) to 40.0% (108/270) when rings share at least two atoms, and from 22.6% (122/540) to 46.7% (28/60) when it contains at least two stereocenters. The Scaffold-OOD results exhibit the same qualitative pattern, with higher structural-failure rates for larger scaffolds and scaffolds containing more stereocenters; the difference associated with greater ring sharing is smaller. Claude-Opus-5.0 also shows higher failure rates in the more complex ID groups (52.8%, 50.4%, and 56.7%, respectively), with the same direction of change on Scaffold-OOD instances.

Errors in structural translation. Among instances for which SREV-Mol produced a final answer, inspection of the SMARTS queries generated during *Translation* identifies errors in 40.3% (54/134) of ID and 26.7% (8/30) of Scaffold-OOD instances containing a biaryl motif. For bridged ring systems on ID instances, the rate rises to 80.0% (12/15), although this subset is small. These translation errors concern the generated query and are distinct from failures measured on the final molecule.

Property failures. For SREV-Mol on ID instances, the largest numbers of individual property-constraint failures occur for QED (59/350), BBBP (56/363), logS (40/300), logD (38/327), and Mutag (35/401). BBBP, logD, logS, and Mutag rely on property predictors, whereas QED combines several molecular descriptors through nonlinear desirability functions. For each of these five properties, we compare the rate of failing at least one requested property constraint between instances that include the property and those that do not, then macro-average the five rates. For SREV-Mol, the mean failure rates with the property present are 26.5% on ID (mean $N = 348$) and 62.3% on Scaffold-OOD (mean $N = 288$), compared with 16.2% (mean $N = 552$) on ID when it is absent. For Claude-Opus-5.0, the corresponding rates are 28.4% on ID (mean $N = 348$) and 68.1% on Scaffold-OOD (mean $N = 288$) when present, versus 14.8% (mean $N = 552$) on ID when absent. Nearly every Scaffold-OOD instance requests each of these five properties (mean $N = 12$ absent), so we do not report an absent-condition rate for that split. On instances involving these five properties, SREV-Mol exhibits a smaller ID-to-Scaffold-OOD increase in the macro-averaged property-failure rate than Claude-Opus-5.0 ($+35.8$ versus $+39.7$ percentage points).

Table 11: Global and auxiliary features used by the edit-selection network. The placeholder <prop> ranges over 14 constrained properties: HBD, HBA, rotatable bonds, total rings, heavy atoms, molecular weight, logP, logD, logS, TPSA, QED, BBBP, mutagenicity, and molar refractivity. The placeholder <fg> ranges over a catalog of 61 functional-group patterns.

Feature	Description
Global Features ($\mathbf{g} \in \mathbb{R}^{88}$)	
st_val__<prop> ($\times 14$)	Measured property value of the current molecule.
st_has_lo__<prop> ($\times 14$)	Indicator that the property has a lower-bound constraint.
st_has_hi__<prop> ($\times 14$)	Indicator that the property has an upper-bound constraint.
st_lo_z__<prop> ($\times 14$)	Signed distance from the lower bound (value minus bound), normalized by the property-specific interquartile range (IQR).
st_hi_z__<prop> ($\times 14$)	Signed distance from the upper bound (bound minus value), normalized by the property-specific IQR.
st_n_props	Number of constrained properties.
st_n_violated	Number of currently violated constraints.
st_gap	Total IQR-normalized distance to the target box.
st_worst_z	Distance of the most severely violated property.
st_depth_left	Remaining edit budget specified by the search harness; unavailable at inference time.
st_heavy	Number of heavy atoms in the current molecule.
st_free_h_sites	Number of atoms bearing a free hydrogen and thus available as attachment sites.
st_n_fg	Number of distinct functional-group types present.
st_n_fg_total	Total number of functional-group occurrences.
st_room_hi_min	Minimum remaining margin to an upper bound; negative values indicate an existing violation.
set_n_cands	Number of applicable candidates at the current node.
set_prob_spread	Range of c_prob across sibling candidates; a narrow range indicates little separation.
h_step_index	Number of edits committed so far.
h_pred_error_last	Error between the predicted and measured property changes for the previous edit.
h_has_history	Indicator that a previous edit exists; zero for the seed molecule.
h_gap_improved_last	Indicator that the previous edit reduced the measured property gap.
Auxiliary Features ($\mathbf{e}_i^{\text{aux}} \in \mathbb{R}^{167}$)	
r_dmean__<prop> ($\times 14$)	Mean predicted property change from MMP statistics, normalized by the property-specific IQR.
r_dstd__<prop> ($\times 14$)	Standard deviation of the predicted property change, representing uncertainty.
r_dfg__<fg> ($\times 61$)	Change in functional-group count ($+1$ for creation and -1 for removal) over the 61-pattern catalog.
c_rank	Rank assigned by the suggestion tool, with zero denoting the top candidate.
c_pred_gap	Predicted distance to the target box after applying the edit.
c_gap_reduction	Predicted gap reduction: current gap minus post-edit gap.
c_n_sat_now	Number of constraints currently satisfied.
c_n_sat_post	Predicted number of constraints satisfied after the edit.
c_dcount	Predicted increase in the number of satisfied constraints.
c_damage	Number of currently satisfied constraints predicted to be violated by the edit.
c_fix_worst	Indicator that the edit improves the most severely violated property.
c_worst_after	Residual violation of the worst property after the edit.
c_cos	Cosine similarity between the predicted property-change vector and the required direction.
c_cover_frac	Fraction of the required movement toward the target covered by the edit.

Continued on next page

Table 11 continued

Feature	Description
c_overshoot	Extent to which the edit crosses the target interval along an intended correction axis.
c_prob	Expected number of satisfied constraints under Gaussian property-change estimates; the only ranking statistic that uses standard deviations.
c_prob_all	Product of per-property satisfaction probabilities, estimating simultaneous satisfaction of all constraints.
c_prob_min	Minimum per-property satisfaction probability.
c_n_props_helped	Number of properties moved toward their target intervals.
c_n_props_hurt	Number of properties moved away from their target intervals.
c_worst_prop_delta	Signed predicted change for the bottleneck property in normalized units.
c_room_frac_used	Fraction of the tightest remaining upper-bound margin consumed by the edit.
c_std_sum	Sum of predicted-change standard deviations over constrained properties.
c_std_max	Maximum predicted-change standard deviation.
c_snr_min	Minimum property-wise ratio |mean change|/standard deviation.
r_is_attach	Indicator of a pure attachment edit (`from = [*:1]`).
r_is_delete	Indicator of a deletion edit (`to = [*:1][H]`).
r_n_cuts	Number of cut points: one, two, or three.
r_heavy_from	Number of heavy atoms in the removed fragment.
r_heavy_to	Number of heavy atoms in the added fragment.
r_dheavy	Change in heavy-atom count.
r_dmw	Change in molecular weight.
r_drings_arom	Change in the number of aromatic rings.
r_drings_aliph	Change in the number of aliphatic rings.
r_drotb	Change in the number of rotatable bonds.
r_dhbd	Change in the number of hydrogen-bond donors.
r_dhba	Change in the number of hydrogen-bond acceptors.
r_dtpsa	Change in topological polar surface area.
r_dlogp	Change in Crippen logP.
r_dhalogen	Change in the number of halogen atoms.
r_dsp3	Change in the fraction of sp^3 carbon atoms.
r_dcharge	Change in formal charge.
r_dsa	Change in synthetic accessibility score.
r_dqed	Fragment-level change in QED.
r_dtpsa_per_heavy	Change in TPSA per heavy atom.
r_dlogp_per_heavy	Change in logP per heavy atom.
r_dhba_per_heavy	Change in HBA count per heavy atom.
r_dpolar_frac	Per-atom change in HBA plus HBD count, representing fragment polarity density.
fg_n_created	Number of functional groups created according to the 61-pattern catalog.
fg_n_destroyed	Number of functional groups removed according to the catalog.
r_is_swap	Indicator of a substitution edit.
site_ok	Indicator that attachment-site computation succeeded.
site_n_anchors	Number of anchors fixed by the edit.
site_aromatic	Indicator that the attachment-site atom is aromatic.
site_in_ring	Indicator that the attachment site belongs to a ring.
site_ring_size	Size of the smallest ring containing the site; zero outside rings.
site_fused	Indicator that the attachment site belongs to a fused ring.
site_is_hetero	Indicator that the attachment-site atom is a heteroatom.
site_degree	Degree of the attachment-site atom.
site_num_h	Number of hydrogens on the attachment-site atom.
site_charge	Gasteiger partial charge of the attachment-site atom.
site_logp_contrib	Crippen logP contribution of the attachment-site atom.
site_sym_equiv	Number of symmetry-equivalent attachment sites.

Continued on next page

Table 11 continued

Feature	Description
site_n_hetero_2b	Number of heteroatoms within two bonds of the attachment site.
site_on_fg	Indicator that the attachment site lies on a functional group.
site_dist_to_fg	Bond distance to the nearest functional group.
site_crowd_2b	Number of heavy atoms within two bonds, measuring local crowding.
site_on_guard	Indicator that the attachment site lies in the protected guard region.
site_dist_to_guard	Bond distance to the guard region, distinguishing core from peripheral sites.
site_frac_guard	Fraction of molecular atoms belonging to the guard region.
ctx_support	Number of MMP pairs supporting the transformation rule.
ctx_log_support	$\log(1 + \text{MMP support})$.
ctx_shift	Magnitude of the candidate-level adjustment to the stored transformation mean.
ctx_fired	Indicator that a candidate-level transformation adjustment is available.
ctx_std_r0	Aggregate dispersion of the transformation's property changes in normalized units.
c_prob_margin	Margin in c_prob over the second-best sibling candidate.
c_prob_z	Standardized c_prob relative to the mean and standard deviation of sibling candidates.
c_is_unique_best	Indicator that the candidate uniquely maximizes c_prob.
cand_similarity	Mean Tanimoto similarity between the candidate fragment and other candidate fragments.
h_motif_repeat	Indicator that the fragment to be attached is already present in the molecule.
h_same_rule_as_prev	Indicator that the candidate repeats the rule used in the previous edit.