Coder as Editor: Code-driven Interpretable Molecular Editing

Wenyu Zhu [1] **Chengzhu Li** [2] **Xiaohe Tian** [3] **Yifan Wang** [4] **Yinjun Jia** [1] **Jianhui Wang** [5] **Bowen Gao** [1 4] **Haichuan Tan** [1 4] **Ya-Qin Zhang** [1] **Wei-Ying Ma** [1] **Yanyan Lan** [1 6]

Abstract

Molecular design is a central task in drug discovery that requires precise structural reasoning and domain knowledge. While large language models (LLMs) have shown promise in generating high-level editing intentions in natural language, they often struggle to faithfully execute these modifications, particularly when operating on non-intuitive representations like SMILES. We introduce MECo, a framework that bridges reasoning and execution by translating editing actions into executable code. MECo reformulates molecular modification for LLMs as a cascaded framework: generating human-interpretable editing intentions from a molecule and design objective, followed by translating those intentions into executable structural edits via code generation. Our approach achieves over 98% accuracy in reproducing held-out realistic edits derived from chemical reactions and target-specific compound pairs. On downstream benchmarks spanning physicochemical properties and target activities, MECo substantially improves consistency by 38-86 percentage points to 90%+ and achieves higher success rates over SMILES-based baselines while preserving structural similarity. By aligning intention with execution, MECo enables consistent, controllable and interpretable molecular design, laying the foundation for high-fidelity feedback loops and collaborative human–AI workflows in drug discovery.

[1]Institute for AI Industry Research, Tsinghua University, Beijing, China [2]Zhili College, Tsinghua University, Beijing, China [3]School of Pharmaceutical Sciences, Peking University, Beijing, China [4]Department of Computer Science and Technology, Tsinghua University, Beijing, China [5]University of Electronic Science and Technology of China, Chengdu, China [6]Beijing Frontier Research Center for Biological Structure, Tsinghua University, Beijing, China. Correspondence to: Yanyan Lan <lanyanyan@air.tsinghua.edu.cn>.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026)

1. Introduction

Recent reasoning-centric large language models (LLMs) have demonstrated notable progress in scientific problem-solving, particularly in programming (Wang et al., 2023), mathematics (Lewkowycz et al., 2022), and chemistry (Zhang et al., 2024). In chemistry, their success on question answering tasks highlights an ability to encode domain knowledge (Hou et al., 2025), but applying this knowledge to tasks like molecular modification, which require both chemical reasoning and precise structural control, remains a challenge.

A growing number of studies have applied LLMs to molecular modification by prompting them to generate optimized molecules in the form of SMILES strings, conditioned on an input structure and a desired property goal (e.g., increased solubility or binding affinity) (Liu et al., 2024a; Ye et al., 2025). While intuitive, this generation paradigm faces major limitations. First, generated SMILES may not always be chemically plausible. Second, even valid outputs may diverge from the design rationale articulated by reasoning LLMs, undermining interpretability and expert trust. Such misalignments hinder human-in-the-loop workflows where reproducibility and verifiability are critical, and break the feedback loop required for iterative model refinement. Moreover, extensive uncontrolled edits may severely compromise synthetic feasibility by disrupting established synthetic routes and limiting the reuse of existing intermediates, thereby increasing the practical cost and difficulty of experimental validation.

At the root of these limitations is a modality mismatch: SMILES is a linearized encoding of molecular graphs, originally designed for compact storage in cheminformatics systems. A single molecule may correspond to many valid SMILES depending on the traversal path, and even small structural edits can cause large, unintuitive changes in the string. In contrast, chemists typically reason over molecular graphs and use graphical tools instead of directly modifying SMILES, making precise control difficult for LLMs.

To bridge this gap, we propose using code as an intermediate representation, a domain where LLMs have already demonstrated strong proficiency. Rather than generating molecular structures directly as SMILES, we reformulate molecular

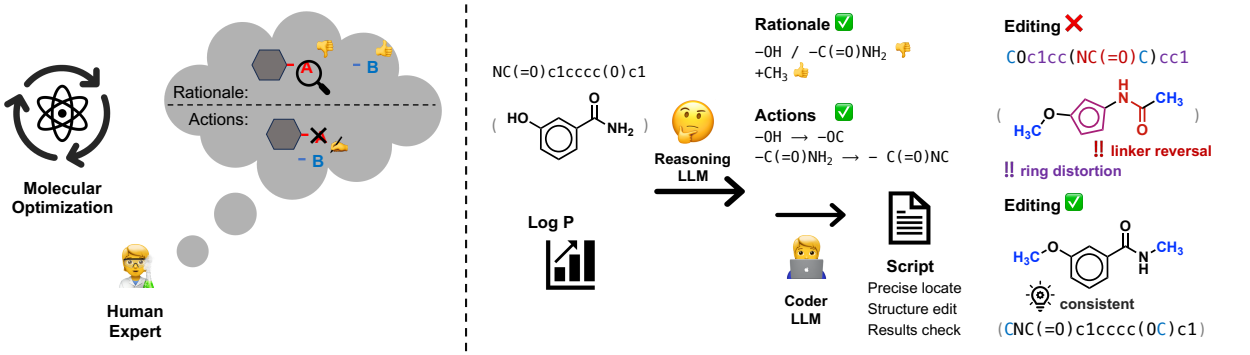

Figure 1. Motivation for MECo: bridging reasoning and execution in molecular optimization through code generation. Left: Human chemists reason over molecular graphs to design targeted edits, such as modifying a functional group to adjust polarity or introducing new interactions, and annotate them directly on the structure. Top right: Reasoning LLMs can generate similar high-level rationales and editing actions, but often struggle to execute them faithfully due to the limitations of sequential molecular representations like SMILES. Bottom right: MECo addresses this gap by introducing a coder LLM that translates editing actions into executable code, enabling precise, interpretable, and reproducible structural edits on the molecular graph.

editing as a code generation task: the LLM produces executable scripts (e.g., using RDKit (Landrum, 2013)) that specify verifiable and interpretable structural modifications. This approach leverages the strengths of LLMs and enables faithful execution of editing intentions to improve controllability, reproducibility, and transparency in molecular design.

We introduce **MECo** (**M**olecular **E**diting via **Co**de generation), a framework that translates high-level design rationales into structured editing programs. We fine-tune a compact code-oriented language model solely on synthetic data generated by limited moiety substitutions on random molecules. The model achieves 98% accuracy on realistic edits derived from reaction and bioactivity datasets, significantly outperforming a same-size general-purpose language model fine-tuned to directly generate SMILES. When using a reasoning LLM such as Deepseek-R1 (Guo et al., 2025) as the upstream component, MECo substantially improves the consistency between editing intentions and resulting structures by 38–86 percentage points to 90%+, leading to higher success rates while preserving structural similarity across multiple property and activity optimization benchmarks.

By bridging natural language reasoning and structural molecular modification through code, MECo enables consistent, controllable and interpretable molecular modification, bringing LLM-guided molecule design closer to real-world scientific application. Our main contributions are:

- **Code-based formulation for molecular modification.** We introduce MECo, a novel framework that recasts molecular editing as a code generation task, enabling LLMs to translate natural language intentions into precise, verifiable, and executable modifications.
- **Scalable data construction for training and evaluation.** We develop a scalable pipeline for constructing both synthetic and realistic editing samples, combining programmatic moiety replacement with edit extraction from chemical reactions and bioactive molecule pairs.
- **Generalization to realistic molecular transformations.** We show that a code LLM trained solely on synthetic edits generalizes effectively to real-world modifications, achieving over 98% accuracy on both reaction- and activity-derived edits.
- **Superior molecule optimization performance.** MECo outperforms direct generation baselines across property and activity benchmarks, with notably *double the structure–intention consistency*, enhancing interpretability and reliability.

2. Related work

Among molecular representations, SMILES (Weininger, 1988) has been widely adopted in sequence-based models, including RNNs (Gómez-Bombarelli et al., 2018; Segler et al., 2018) and Transformers (Schwaller et al., 2019). In contrast, graph-based approaches (Gilmer et al., 2017; Jin et al., 2018a; Shi et al., 2020) and 3D-aware models (Schütt et al., 2017; Satorras et al., 2021) aim to better capture structural validity by operating directly on molecular graphs and spatial coordinates. Large-scale pretraining further produced molecular language models such as ChemBERTa (Chithrananda et al., 2020), MolBERT (Fabian et al., 2020), Chemformer (Irwin et al., 2022), and MolXPT (Liu et al., 2023), which perform competitively with graph-based approaches. However, SMILES poorly align with natural language: small structural edits can cause large string differences, and multiple encodings exist for the same molecule (Merz Jr et al., 2020), making them suboptimal for reasoning in LLMs.

Graph-based molecular optimization has been extensively studied, including gradient-based optimization (Jin et al., 2018a), RL-based editing (Jin et al., 2020b; Shi et al., 2020),, low-to-high motif translation (Jin et al., 2018b; 2020a), and guided diffusion (Vignac et al., 2022). These approaches are powerful but fundamentally rely on oracle-driven optimization(Gao et al., 2022), using predictors, learned surrogates, or reward signals to characterize the underlying property landscape. More recently, reasoning LLMs have introduced a complementary, zero-oracle perspective, where edits can be proposed directly from embedded chemical knowledge without querying task-specific oracles. This opens opportunities for scenarios where oracles are unavailable (e.g., new targets) or expensive (e.g., experimental endpoints), and can naturally interface with existing oracle-based optimizers or language–graph alignment approaches to further expand the molecular design space.

LLMs have been investigated as general-purpose optimizers (Yang et al., 2023; Meyerson et al., 2024; Liu et al., 2024b), and these ideas have recently been extended to molecular domains. Prompt-based approaches such as MOLLEO (Wang et al., 2024) and ChatDrug (Liu et al., 2024a) adapt LLMs to propose molecular modifications, either by embedding them in genetic algorithms or by augmenting them with retrieval databases. Other methods rely on representation learning, for example by exploiting pretrained LLM embeddings (Ranković & Schwaller, 2023) or by fine-tuning general-purpose models on molecular corpora to improve generation quality (Bedrosian et al., 2024; Fang et al., 2023; Kristiadi et al., 2024). DrugAssist (Ye et al., 2025) further contributed a large-scale MolOpt-Instructions dataset to support instruction-tuned optimization models. LICO (Nguyen & Grover, 2024) proposed a semi-synthetic training framework that extends general-purpose LLMs into surrogate models for black-box molecular optimization. MolReasoner (Zhao et al., 2025) introduces a two-stage framework that integrates synthetic Chain-of-Thought supervision with reinforcement learning, shifting molecular LLMs from memorization toward interpretable reasoning. Despite these advances, most existing systems still operate directly in SMILES or graph spaces, which limits their alignment with natural language reasoning and hinders the interpretability of the generated modifications.

3. Methods

3.1. Problem formulation

Molecular optimization is a central task in drug discovery, where the objective is to generate a modified molecule M_o from an initial compound M_i to improve one or more target properties T (e.g., permeability, target binding affinity), while preserving essential structural features such as the core scaffold or pharmacophores, and enabling reuse of steps in a common synthetic route.

In many prior approaches, particularly those based on RNNs or early LLMs, this task is formulated as:

$$M_o = \mathcal{F}(M_i, T) \tag{1}$$

where $\mathcal{F}$ is a black-box model that directly maps the input molecule and target to a SMILES string.

While this formulation has shown some empirical success, it suffers from two key limitations. First, the editing process is entirely implicit: the model does not explain what was changed or why, making the transformation uninterpretable. Second, the output often diverges from well-established medicinal chemistry principles. These models tend to make broad, unconstrained modifications, rather than the minimal, targeted edits that chemists use, such as modifying a functional group to adjust polarity for better permeability, or introducing an H-bond donor to improve affinity and selectivity. Without such structure–property reasoning, the results are hard to interpret or trust, limiting their practical utility in real-world design workflows.

In contrast, recent reasoning-centric LLMs offer a fundamentally different capability: rather than directly generating M_o, they can first articulate a structured editing intention based on M_i and T. These intentions often consist of: **a set of editing actions** $A = \{a_1, a_2, \dots\}$ (e.g., replace the para-methyl with a hydroxyl group), and **a corresponding rationale** (e.g., to increase polarity and improve solubility).

This shift from black-box generation to interpretable, rationale-driven editing actions more closely reflects how medicinal chemists reason about molecular design based on structural and physicochemical considerations. The corresponding formulation becomes:

$$(A, M_o) \leftarrow \mathcal{F}(M_i, T) \tag{2}$$

However, in current systems, this promising capability remains underutilized. While reasoning LLMs can propose chemically meaningful edits, they often fail to execute them faithfully. This is largely due to the difficulty of performing precise and constrained modifications on SMILES representations, which are sensitive to minor syntax errors and lack structural locality. As a result, the generated molecule M_o may deviate from the proposed intention or violate chemical constraints.

To bridge this gap between reasoning and execution, we reformulate molecular editing as a code generation task:

$$\mathcal{C} = \mathcal{G}(M_i, A), \quad M_o = \mathcal{C}(M_i) \tag{3}$$

where $\mathcal{G}$ denotes the code generation model (e.g., a coder LLM), and $\mathcal{C}$ is an executable script (e.g., using RDKit) that implements the specified editing actions A_i.

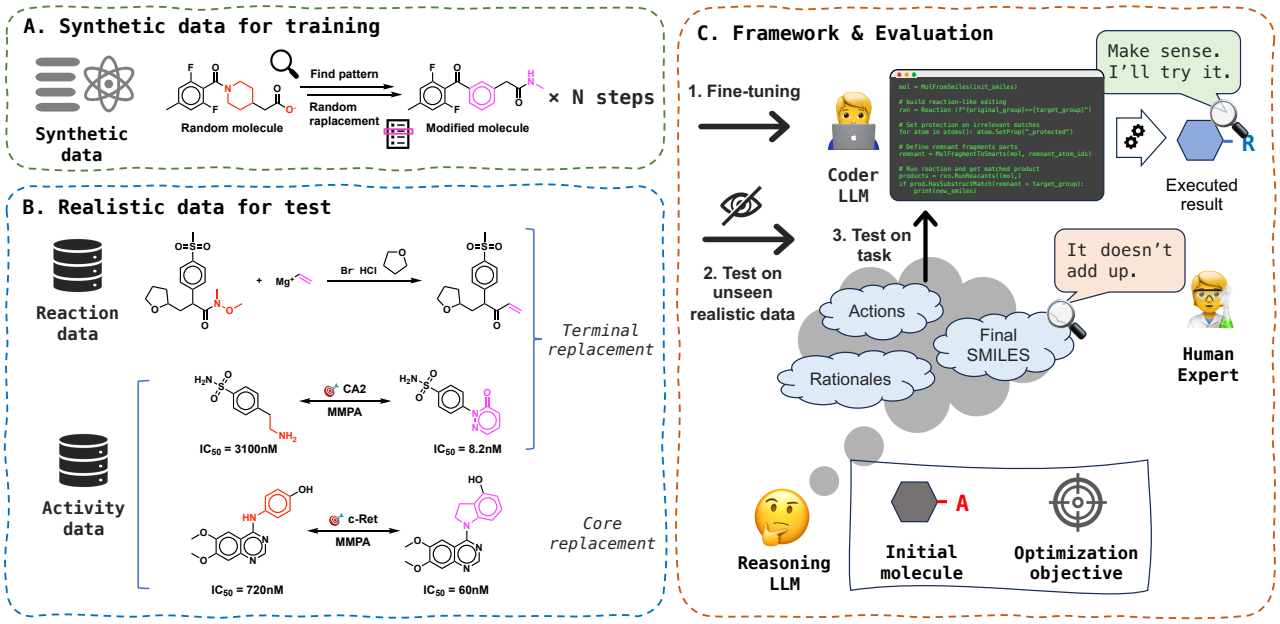

Figure 2. Overview of the MECo framework. (A) Synthetic data construction for model training. (B) Realistic data for testing on unseen molecular modifications derived from both reaction and activity data, reflecting chemical transformations observed in practice and in human reasoning. Modifications are categorized into terminal or core replacements. (C) End-to-end workflow and evaluation of the framework. The coder LLM is evaluated both independently and within the integrated pipeline. A reasoning LLM first generates rationales and editing actions, which are then executed by the coder model to produce optimized molecules. Outputs are subsequently reviewed by human experts to assess the alignment between generated structures and intended editing actions.

Why code as an interface? LLMs have demonstrated strong performance in translating natural language into structured domain-specific languages such as programming code (Fang et al., 2024). Models like CodeT5+ (Wang et al., 2023) and StarCoder (Li et al., 2023) can reliably generate and edit code given natural language instructions. This success is attributed to the syntactic and semantic regularity of code, and structure-aware strategies such as abstract syntax trees (Wang et al., 2021), fill-in-the-middle training (Li et al., 2023), and execution-based feedback (Li et al., 2022).

This motivates us to treat molecular editing as a code generation task. Rather than generating fragile SMILES strings, the LLM produces executable scripts that specify structural edits. Notably, cheminformatics libraries like RDKit (Landrum, 2013) provide robust APIs for manipulating molecular graphs, allowing SMILES strings to be parsed into graph-based data structures and modified programmatically. This enables verifiable and reproducible molecular transformations grounded in chemical structure, not string syntax.

3.2. Framework overview

We propose **MECo** (**M**olecular **E**diting via **Co**de generation), a framework that decouples high-level chemical reasoning from low-level molecular editing, as illustrated in Figure 2C. Given an initial molecule M_i and a target property goal T, a reasoning LLM first generates an editing intention $(A, \text{rationale})$, where A denotes a set of editing actions and the rationale explains their purpose. The focus shifts to executing A through code generation, producing the optimized molecule M_o.

MECo operates in a cascading manner:

Intention generation (reasoning LLM): Generates editing actions A and associated rationales based on (M_i, T).

Action execution (code LLM): Translates A into code $\mathcal{C}$, which is applied to M_i to yield M_o.

$$\underbrace{(M_i, T) \longrightarrow (A,\ \text{rationale})}_{\text{reasoning LLM}} \longrightarrow \underbrace{\mathcal{C} = \text{CodeGen}(A)}_{\text{code LLM}} \longrightarrow M_o = \mathcal{C}(M_i) \tag{4}$$

This decoupled formulation offers several advantages: (1) It separates reasoning (*why* to edit) from execution (*how* to edit), mirroring expert design workflows. (2) It provides interpretable, verifiable, and debuggable records of molecular transformation. (3) It ensures structural validity and fine-grained control by grounding edits in explicitly defined procedures.

By translating natural language intentions into code-level structural edits, MECo enable LLM-driven molecule design that is not only intelligent, but also reliable, transparent, and practically applicable to real-world drug discovery.

3.3. Data construction

To train the code LLM to translate editing actions into executable molecular transformations, we construct a large-scale dataset of molecule–edit–code triples in the form:

$$(M_i, A) \longrightarrow \mathcal{C} \tag{5}$$

where M_i is the input molecule, A is a set of editing actions expressed in natural language, and $\mathcal{C}$ is the corresponding Python code that applies the edit using cheminformatics toolkits such as RDKit.

We adopt a hybrid data construction strategy combining synthetic and realistic samples:

Synthetic edits. We constructed synthetic editing examples by applying programmatically defined moiety replacements to molecules randomly sampled from the ZINC database (Sterling & Irwin, 2015), as shown in Figure 2A. Each initial molecule M_i was iteratively edited by substituting one of its substructures with a moiety from a predefined pattern pool $\mathcal{P} = \bigcup_i \mathcal{P}_i$, where each subset $\mathcal{P}_i$ contains fragments with i attachment points.

At each iteration, a pattern p was randomly drawn from the pool, and a matching site in M_i was identified. If a match was found, it was replaced with another randomly selected moiety $r \in \mathcal{P}_i \setminus \{p\}$ of the same connectivity. If no match was found, the process continued with a different pattern until the pool was exhausted. Replacement was performed iteratively, with all successful edits recorded. For each edit, the corresponding executable code snippet $\mathcal{C}$ was automatically generated, and the edited molecule M_o was obtained by applying $\mathcal{C}$ to the input molecule M_i, i.e., $M_o = \mathcal{C}(M_i)$.

This procedure yields chemically valid and structurally controlled synthetic edits, providing precise edit-code supervision that enables the model to learn robust mappings from actions to code without requiring exposure to real-world editing data. The full algorithm and list of predefined patterns are provided in Appendix A.1.

Realistic edits. To evaluate generalization beyond synthetic edits, we construct a set of realistic editing samples derived from two sources (Figure 2B):

(1) Reaction-derived edits: We extracted matched reactant–product pairs from the USPTO-MIT reaction dataset (Jin et al., 2017). Atom mapping was used to identify structural changes, and we retained samples with a single modification site while preserving the molecular core.

(2) Target-based edits: We extracted compounds from the ChEMBL35 database (Mendez et al., 2018) and grouped them by target ID after applying basic structure-based filters (details provided in Appendix A.2). Matched molecular pairs (MMPs) were identified using the algorithm by Hussain & Rea (2010), and further filtered following the criteria described in Appendix A.3. Attachment points were explicitly identified and recorded to enable precise modification tracking. Based on the number of attachment points in each transformation, the data were categorized into two types: **terminal replacements** (single-point attachment, same as those in **Reaction-derived edits**) and **core replacements** (multi-point attachment).

Together, these two sources reflect the two most common strategies in real-world molecular design. The first captures reaction-derived transformations, grounded in feasible chemical synthesis steps. The second comprises target-specific structural modifications observed across bioactive compounds, which represent plausible edits made during lead optimization. While the directionality of property change is not explicitly defined in our setup, these samples provide chemically meaningful, human-curated edits. Although our model is trained solely on synthetic data, evaluation on these realistic edits allows us to assess its generalization to authentic molecular transformation scenarios.

3.4. Model Training

To train the coder LLM for accurate molecular editing, we used a curated dataset of 50,000 synthetic editing samples (Section 3.3). Each sample includes a molecule M_i, an editing action A described using a combination of natural language and cheminformatics notation (e.g., SMARTS), and a Python code snippet $\mathcal{C}$ that transforms M_i accordingly. For training the direct SMILES generation baseline, the corresponding output molecules $M_o = \mathcal{C}(M_i)$ were used as targets. All models were fine-tuned using the official Qwen2.5-Coder finetuning framework. Additional details regarding prompt design, formatting choices, and training settings are provided in Appendix B.

3.5. Benchmark and evaluation metrics

To evaluate both the core editing capability and downstream utility of our framework, we introduced two complementary benchmarks aligned with our problem formulation:

1. Molecular editing benchmark for problem diagnosis. We used the realistic editing samples constructed in Section 3.3 to identify the limitations of existing approaches and to quantitatively evaluate the molecular editing capabilities of MECo. For each sample, we assess whether the model can faithfully execute the specified editing action A on the input molecule M_i, either directly or via code generation, to produce the expected output molecule M_o.

2. Molecular modification benchmark for application endpoint. To assess the practical utility of our framework, we evaluate end-to-end performance on ChemCoTBench (Li et al., 2025a). This benchmark comprises six molecular

Table 1. Execution accuracy (%) across realistic edit benchmarks. Rows are shaded to distinguish model variants: Qwen2.5, Qwen2.5-Coder, and finetuned versions (darker, denoted by -FT).

Action	Model	Terminal replacement		Core replacement
		Reaction-derived	Target-specific	Target-specific
Direct	Qwen2.5-7B-Instruct	0.5	1.3	0.0
	Qwen2.5-7B-Instruct-FT	38.7	7.4	2.7
CodeGen	Qwen2.5-7B-Instruct	35.4	3.7	4.1
	Qwen2.5-Coder-7B-Instruct	50.3	9.9	12.2
	Qwen2.5-Coder-7B-Instruct-FT (ours)	**99.9**	**98.3**	**98.3**

modification tasks: three involving physicochemical properties (penalized logP (Gómez-Bombarelli et al., 2018), solubility (Delaney, 2004), QED (Bickerton et al., 2012), implemented in RDKit (Landrum, 2013)) and three involving target-specific bioactivities (DRD2 (Olivecrona et al., 2017), JNK3, and GSK3β (Li et al., 2018), implemented in TDC Oracles (Huang et al., 2021)). Each task provides a set of input molecules M_i paired with a shared optimization goal T. This setting reflects real-world design scenarios, where high-level property goals must be translated into concrete molecular modifications. For each (M_i, T) pair, we compare direct molecule generation from a reasoning LLM with the MECo framework, which first generates an editing intention and then executes it via code. This evaluation allows us to assess whether MECo improves over direct generation approaches in terms of (1) chemical validity, (2) optimization success rate, (3) mean similarity to source molecules, and (4) structure–action consistency, which is computed only for cases with human-interpretable edits.

4. Experiments

4.1. Experimental setup

In the molecular modification task under the MECo framework, we applied a unified wrapper to the original prompt to constrain the output format and facilitate reliable extraction of editing actions. The full wrapper template is provided in Appendix C.2. To obtain ground-truth labels, we manually applied the actions generated by the reasoning LLMs to the initial molecules, following the evaluation criteria described in Appendix C.3.

We selected DeepSeek-R1 (Guo et al., 2025) as the reasoning LLM used in MECo by default, due to its strong performance in structured scientific reasoning and its fully open-source availability, which facilitates reproducibility and downstream integration. Gemini-2.5-Pro (Comanici et al., 2025) was also evaluated for comparison, given its top performance on ChemCoTBench. We employed Qwen2.5-7B (Yang et al., 2025) as the edit execution model, fine-tuning its Coder variant (Hui et al., 2024) for structured code-based editing, and its general-purpose variant for direct SMILES generation.

4.2. Edit execution on realistic benchmark

To ensure a fair evaluation of editing generalization, we removed all samples from each realistic dataset whose original or modified fragments had a Tanimoto similarity ≥ 0.6 (computed using ECFP4 (Rogers & Hahn, 2010)) with any moiety used in the synthetic training set. From the remaining pool, we randomly sampled 1,000 examples per category: reaction-derived, target-specific terminal replacement, and target-specific core replacement, as the test sets. This setup ensures that the model is evaluated on structurally diverse and unseen edits, providing a robust assessment of its generalization ability. Results are summarized in Table 1. Appendix Figure 6 further visualizes the fragment distributions via t-SNE, showing the broad and largely non-overlapping chemical space of realistic edits.

This comparison demonstrates the advantage of code-based molecular editing over direct SMILES generation, independent of both string and structural similarities (see Appendix Figure 7). The direct generation model Qwen2.5-7B-Instruct exhibits extremely low execution accuracy across all categories, suggesting a limited ability to faithfully apply structural modifications. Notably, even without specialized training, prompting the same general LLM to generate code leads to substantially higher accuracy than its direct generation counterpart, underscoring the inherent advantages of structured code over direct SMILES generation. Interestingly, we find that the discrepancy between reaction-derived and target-specific terminal replacements can be largely attributed to differences in SMILES syntax patterns, suggesting that LLMs struggle to generalize over SMILES syntax in both direct generation and code generation. A detailed discussion is provided in Appendix C.4.

With supervised fine-tuning, the code-specialized model Qwen2.5-Coder-7B-Instruct-FT achieves over 98% execution accuracy across all benchmarks, while direct generation shows only marginal improvement. These results affirm that MECo's code generation formulation dramatically improves execution fidelity, even on realistic and diverse editing tasks.

Table 2. Performance on molecular optimization tasks. Each task reports validity rate (VR%), success rate (SR%), mean similarity to the source molecule (Sim), and consistency rate (CR%). **Bold** numbers highlight the best value in each column. Note: GPT-5 refused to answer a small number of samples.

Property optimization tasks												
Model	**Penalized logP**				**Solubility**				**QED**			
	VR%	SR%	Sim	CR%	VR%	SR%	Sim	CR%	VR%	SR%	Sim	CR%
W/o thinking												
GPT-4o	22	8	0.11	–	41	35	0.21	–	31	17	0.11	–
Gemini-2.0-flash	66	44	0.27	–	32	28	0.13	–	64	54	0.26	–
DeepSeek-V3	53	21	0.20	–	52	44	0.21	–	48	33	0.18	–
W/ thinking												
GPT-5	58	32	0.21	–	67	63	0.25	–	68	66	0.21	–
Gemini-2.5-Pro	78	69	0.37	45	79	79	0.40	42	83	**83**	0.33	55
DeepSeek-R1	62	48	0.31	30	70	64	0.39	28	68	61	0.31	29
MECo	**93**	**72**	**0.54**	**96**	**96**	**96**	**0.63**	**95**	**87**	75	**0.49**	**95**
Activity optimization tasks												
Model	**DRD2**				**JNK3**				**GSK3β**			
	VR%	SR%	Sim	CR%	VR%	SR%	Sim	CR%	VR%	SR%	Sim	CR%
W/o thinking												
GPT-4o	27	12	0.12	–	20	4	0.08	–	19	10	0.08	–
Gemini-2.0-flash	60	32	0.29	–	56	17	0.25	–	71	39	0.36	–
DeepSeek-V3	46	22	0.20	–	40	9	0.14	–	43	18	0.15	–
W/ thinking												
GPT-5	64	50	0.20	–	57	15	0.17	–	46	28	0.14	–
Gemini-2.5-Pro	85	64	0.36	43	74	24	0.32	42	74	51	0.38	29
DeepSeek-R1	73	54	0.36	33	43	8	0.19	12	45	28	0.22	20
MECo	**89**	**66**	**0.55**	**93**	**92**	**39**	**0.55**	**98**	**91**	**60**	**0.56**	**96**

4.3. Improvement on molecular modification

Table 2 summarizes performance across six molecular optimization tasks, covering both physicochemical properties and target activities. We compare several non-reasoning / reasoning LLMs with our proposed MECo framework, which augments reasoning-centric molecular modification with code generation for edit execution. We omit mean property improvement from the main results, since models can achieve large gains by sacrificing structural preservation and producing molecules that differ substantially from the initial ones. Such cases inflate the average improvement without reflecting meaningful optimization. We therefore adopt success rate, and provide the mean improvement results in Appendix C.5.

MECo improves success rates (SR%) while maintaining higher structural similarity (Sim), suggesting more effective yet conservative edits. Importantly, it shows markedly higher consistency rate (CR%), which measure alignment between output structures and editing intentions, highlighting its faithfulness to reasoning rationales. This alignment enhances interpretability by ensuring that the optimized molecules reflect the reasoning LLM's intended actions, resulting in more consistent and transparent modification that can be readily understood and verified by human experts.

In contrast, direct SMILES generation often produces unintended or overly disruptive modifications, with poor alignment to the specified editing instructions (see cases in Section 4.4). Even strong reasoning LLMs fall short when lacking explicit action execution, underscoring the need to bridge high-level reasoning and low-level structural manipulation through code-based editing.

Figure 3 further shows that MECo's improvements generalize across reasoning LLMs (DeepSeek-R1 and Gemini-2.5-Pro), demonstrating its broad applicability and utility in controllable, interpretable molecule design.

4.4. Case study

To illustrate the difference between direct generation and our MECo framework, we analyze molecular modification tasks and present representative cases in Figure 4A. Direct generation often introduces uncontrolled structural changes

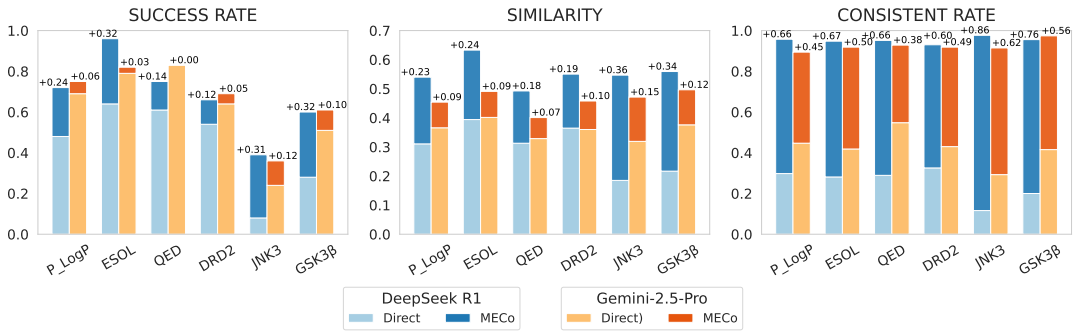

Figure 3. Incremental bar plots comparing direct generation (light bars) and MECo (dark stacked bars) across tasks and metrics under two different reasoning LLMs. Numbers above the bars indicate the relative improvements of MECo over direct.

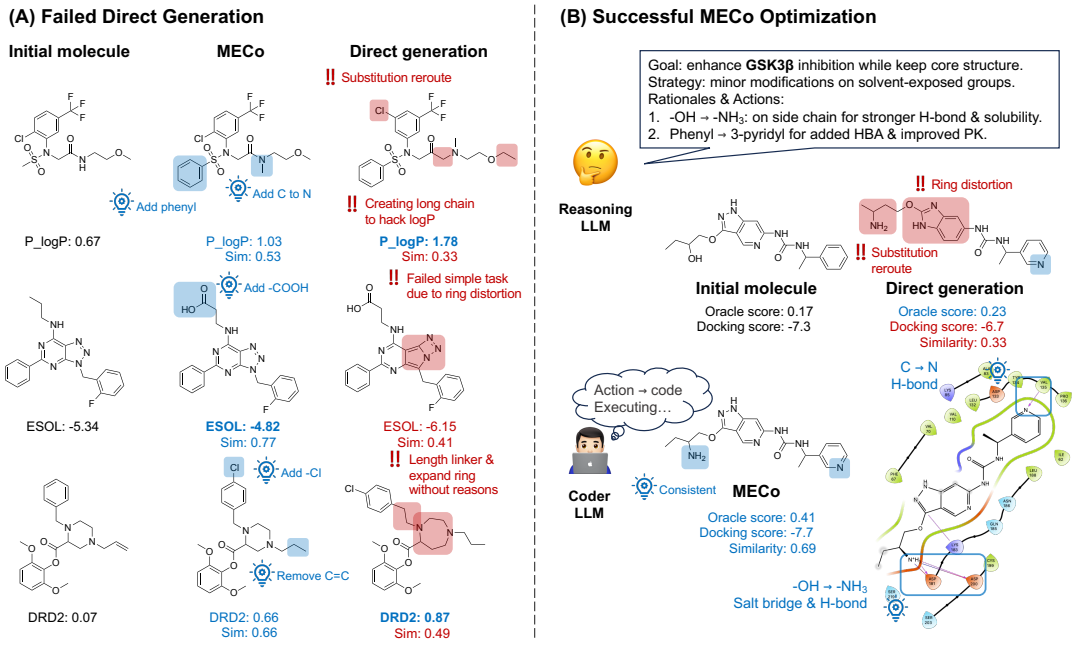

Figure 4. Representative case study of molecular modification. (A) Direct generation produces uncontrolled changes and low similarity, often leading to unrealistic score inflation or failed optimization; MECo achieves improved scores and higher similarity through interpretable, chemistry-preserving edits. (B) MECo demonstrates coherent consistency across the reasoning–execution–analysis pipeline.

with low similarity to the initial molecule, leading to unrealistic score inflation or failed optimization. In contrast, MECo achieves improved scores while maintaining high similarity through interpretable, chemistry-preserving edits.

The detailed example on GSK3β activity optimization task in Figure 4B further demonstrates how MECo maintains consistency across the reasoning–execution–analysis pipeline. The reasoning LLM proposes rational modifications, such as replacing a hydroxy with an amino to strengthen hydrogen bonding and improve solubility, and substituting a phenyl ring with a pyridin-3-yl ring to introduce an additional hydrogen bond acceptor and enhance pharmacokinetic potential. The coder LLM then faithfully translates these actions into executable edits, ensuring that the intended modifications are precisely applied. This yields improved oracle and docking scores, while complementary binding analysis further validates the rationales. The coherent consistency underpins the interpretability and reliability of MECo compared to direct generation.

5. Conclusion

We introduce MECo, a framework that reformulates molecular modification as a code generation task, bridging high-level reasoning with low-level structural execution. By translating interpretable editing intentions into executable scripts, MECo achieves near-perfect edit fidelity on realistic benchmarks and consistently outperforms direct-SMILES-generation baselines across both physicochemical property and bioactivity optimization tasks. This formulation enables

consistent, controllable and interpretable molecular modification workflows, paving the way for trustworthy human-AI collaboration and high-fidelity feedback loops between AI models and experimental validation in drug discovery, such as agent-style extensions and multi-round optimization. Beyond this application, our results suggest a general paradigm for bridging reasoning and execution in scientific domains, highlighting the potential of LLMs as reliable assistants for structured, verifiable discovery workflows.

Acknowledgment

This work is supported by Innovative Drug Research and Development-National Science and Technology Major Project (No.2025ZD1802501) and Beijing Frontier Research Center for Biological Structure Fundings.

Impact Statement

This work advances machine learning for interpretable molecular modification by leveraging natural language reasoning to mimic expert chemist thinking. Rather than fitting property or activity landscapes, it points toward human-AI collaboration to explore unobserved areas, particularly for underexplored targets with scarce data. Future applications may integrate with automated experimental workflows to systematically investigate new domains without overfitting limited datasets, supporting controlled and explainable molecular discovery.

References

Bedrosian, M., Guevorguian, P., Fahradyan, T., Chilingaryan, G., Khachatrian, H., and Aghajanyan, A. Small molecule optimization with large language models. In *Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges*, 2024.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, February 2012. ISSN 1755-4349. doi: 10.1038/nchem.1243. URL https://www.nature.com/articles/nchem.1243. Number: 2 Publisher: Nature Publishing Group.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities, July 2025. URL http://arxiv.org/abs/2507.06261. arXiv:2507.06261 [cs].

Delaney, J. S. ESOL: Estimating Aqueous Solubility Directly from Molecular Structure. *ACS Publications*, March 2004. doi: 10.1021/ci034243x. URL https://pubs.acs.org/doi/abs/10.1021/ci034243x. Archive Location: world Publisher: American Chemical Society.

Fabian, B., Edlich, T., Gaspar, H., Segler, M., Meyers, J., Fiscato, M., and Ahmed, M. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*, 2020.

Fang, C., Miao, N., Srivastav, S., Liu, J., Zhang, R., Fang, R., Asmita, Tsang, R., Nazari, N., Wang, H., and Homayoun, H. Large language models for code analysis: do LLMs really do their job? In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, pp. 829–846, USA, August 2024. USENIX Association. ISBN 978-1-939133-44-1.

Fang, Y., Zhang, N., Chen, Z., Guo, L., Fan, X., and Chen, H. Domain-agnostic molecular generation with self-feedback. *arXiv preprint arXiv:2301.11259*, 2023.

Gao, W., Fu, T., Sun, J., and Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z. Publisher: Nature Publishing Group.

Hou, Y., Zhan, Z., and Zhang, R. Benchmarking GPT-5 for biomedical natural language processing, August 2025. URL http://arxiv.org/abs/2509.04462. arXiv:2509.04462 [cs].

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C., Xiao, C., Sun, J., and Zitnik, M. Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, December 2021.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Dang, K., et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Hussain, J. and Rea, C. Computationally Efficient Algorithm to Identify Matched Molecular Pairs (MMPs) in Large Data Sets. *Journal of Chemical Information and Modeling*, 50(3):339–348, March 2010. ISSN 1549-9596. doi: 10.1021/ci900450m. URL https://doi.org/10.1021/ci900450m. Publisher: American Chemical Society.

Irwin, R., Dimitriadis, S., He, J., and Bjerrum, E. J. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

Jin, W., Coley, C. W., Barzilay, R., and Jaakkola, T. Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network, December 2017. URL http://arxiv.org/abs/1709.04555. arXiv:1709.04555 [cs].

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018a.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018b.

Jin, W., Barzilay, D. R., and Jaakkola, T. Hierarchical Generation of Molecular Graphs using Structural Motifs. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 4839–4848. PMLR, November 2020a. URL https://proceedings.mlr.press/v119/jin20a.html. ISSN: 2640-3498.

Jin, W., Barzilay, D. R., and Jaakkola, T. Multi-Objective Molecule Generation using Interpretable Substructures. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 4849–4859. PMLR, November 2020b. URL https://proceedings.mlr.press/v119/jin20b.html. ISSN: 2640-3498.

Kristiadi, A., Strieth-Kalthoff, F., Skreta, M., Poupart, P., Aspuru-Guzik, A., and Pleiss, G. A sober look at llms for material discovery: Are they actually good for bayesian optimization over molecules? *arXiv preprint arXiv:2402.05015*, 2024.

Landrum, G. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8, 2013.

Lewkowycz, A., Andreassen, A. J., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V. V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Li, H., Cao, H., Feng, B., Shao, Y., Tang, X., Yan, Z., Yuan, L., Tian, Y., and Li, Y. Beyond Chemical QA: Evaluating LLM's Chemical Reasoning with Modular Chemical Operations, June 2025a. URL http://arxiv.org/abs/2505.21318. arXiv:2505.21318 [cs].

Li, M., Li, H., and Tan, C. HypoEval: Hypothesis-Guided Evaluation for Natural Language Generation, April 2025b. URL http://arxiv.org/abs/2504.07174. arXiv:2504.07174 [cs].

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Li, Y., Zhang, L., and Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics*, 10(1):33, July 2018. ISSN 1758-2946. doi: 10.1186/s13321-018-0287-6. URL https://doi.org/10.1186/s13321-018-0287-6.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A. D., Hubert, T., Choy, P., d'Autume, C. d. M., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Robson, E. S., Kohli, P., Freitas, N. d., Kavukcuoglu, K., and Vinyals, O. Competition-level code generation with AlphaCode. *Science*, December 2022. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/10.1126/science.abq1158. Publisher: American Association for the Advancement of Science.

Lin, Y.-T. and Chen, Y.-N. LLM-Eval: Unified Multi-Dimensional Automatic Evaluation for Open-Domain Conversations with Large Language Models. In Chen, Y.-N. and Rastogi, A. (eds.), *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pp. 47–58, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlp4convai-1.5. URL https://aclanthology.org/2023.nlp4convai-1.5/.

Liu, S., Wang, J., Yang, Y., Wang, C., Liu, L., Guo, H., and Xiao, C. Conversational drug editing using retrieval and domain feedback. In *The twelfth international conference on learning representations*, 2024a.

Liu, T., Astorga, N., Seedat, N., and van der Schaar, M. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024b.

Liu, Z., Zhang, W., Xia, Y., Wu, L., Xie, S., Qin, T., Zhang, M., and Liu, T.-Y. Molxpt: Wrapping molecules with text for generative pre-training. *arXiv preprint arXiv:2305.10688*, 2023.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C. J., Segura-Cabrera, A., Hersey, A., and Leach, A. R. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 2018. ISSN 0305-1048. doi: 10.1093/nar/gky1075.

Merz Jr, K. M., De Fabritiis, G., and Wei, G.-W. Generative models for molecular design, 2020.

Meyerson, E., Nelson, M. J., Bradley, H., Gaier, A., Moradi, A., Hoover, A. K., and Lehman, J. Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning*, 4(4):1–40, 2024.

Nguyen, T. and Grover, A. Lico: Large language models for in-context molecular optimization. *arXiv preprint arXiv:2406.18851*, 2024.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, September 2017. ISSN 1758-2946. doi: 10.1186/s13321-017-0235-x. URL https://doi.org/10.1186/s13321-017-0235-x.

Ranković, B. and Schwaller, P. Bochemian: Large language model embeddings for bayesian optimization of chemical reactions. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Rogers, D. and Hahn, M. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, May 2010. ISSN 1549-9596. doi: 10.1021/ci100050t. URL https://doi.org/10.1021/ci100050t. Publisher: American Chemical Society.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8 (1):13890, 2017.

Schwaller, P., Laino, T., Gaudin, T., Bolgar, P., Hunter, C. A., Bekas, C., and Lee, A. A. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4 (1):120–131, 2018.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Sterling, T. and Irwin, J. J. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, November 2015. ISSN 1549-9596. doi: 10.1021/acs.jcim.5b00559.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. DiGress: Discrete Denoising diffusion for graph generation. September 2022. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Wang, H., Skreta, M., Ser, C.-T., Gao, W., Kong, L., Strieth-Kalthoff, F., Duan, C., Zhuang, Y., Yu, Y., Zhu, Y., et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024.

Wang, Y., Wang, W., Joty, S., and Hoi, S. C. H. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation, September 2021. URL http://arxiv.org/abs/2109.00859. arXiv:2109.00859 [cs].

Wang, Y., Le, H., Gotmare, A. D., Bui, N. D., Li, J., and Hoi, S. C. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*, 2023.

Weininger, D. 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.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 Technical Report, January 2025. URL http://arxiv.org/abs/2412.15115. arXiv:2412.15115 [cs].

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Ye, G., Cai, X., Lai, H., Wang, X., Huang, J., Wang, L., Liu, W., and Zeng, X. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26 (1):bbae693, 2025.

Ye, S., Kim, D., Kim, S., Hwang, H., Kim, S., Jo, Y., Thorne, J., Kim, J., and Seo, M. FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets. October 2023. URL https://openreview.net/forum?id=CYmF38ysDa.

Zhang, D., Liu, W., Tan, Q., Chen, J., Yan, H., Yan, Y., Li, J., Huang, W., Yue, X., Ouyang, W., et al. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024.

Zhao, G., Li, S., Lu, Z., Cheng, Z., Lin, H., Wu, L., Xia, H., Cai, H., Guo, W., Wang, H., et al. Molreasoner: Toward effective and interpretable reasoning for molecular llms. *arXiv preprint arXiv:2508.02066*, 2025.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. November 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-Following Evaluation for Large Language Models, November 2023. URL http://arxiv.org/abs/2311.07911. arXiv:2311.07911 [cs].

A. Data construction details

A.1. Synthetic edit generation algorithm

We provide here the full algorithm for generating synthetic editing examples, as described in Section 3.3. The procedure applies pre-defined moiety replacements to randomly sampled molecules from the ZINC database, using structured pattern pools grouped by attachment point count.

Algorithm 1 Iterative Moiety Replacement

Require: Molecule M_i, pattern pool $\mathcal{P} = \bigcup_i \mathcal{P}_i$, iterations N
$M \leftarrow M_i$ {Initialize molecule}
for $n = 1$ to N **do**
  $\mathcal{P}_{\text{avail}} \leftarrow \mathcal{P}$ {Reset available patterns}
  $replaced \leftarrow$ False
  while $\mathcal{P}_{\text{avail}} \neq \emptyset$ and $replaced =$ False **do**
    Sample $p \sim \mathcal{P}_{\text{avail}}$; $\mathcal{P}_{\text{avail}} \leftarrow \mathcal{P}_{\text{avail}} \setminus \{p\}$ {Draw and remove pattern}
    $\mathcal{S} \leftarrow$ GETSUBSTRUCTMATCHES(M, p) {Identify all matching substructures}
    if $\mathcal{S} \neq \emptyset$ **then**
      $s \sim \mathcal{S}$ {Randomly select a match site}
      $i \leftarrow$ attachment_count(p) {Determine connection type}
      $r \sim \mathcal{P}_i \setminus \{p\}$ {Sample replacement with same connectivity}
      for a in M **do**
        if $a \notin s$ **then**
          Protect(a) {Protect atoms not in the selected substructure match s}
        end if
      end for
      $M \leftarrow$ replace(M, p, r) {Apply replacement using REACTION}
      Record (p, s, r) {s uniquely specifies the matched substructure instance}
      $replaced \leftarrow$ True
    end if
  end while
  if $replaced =$ False **then**
    break {Stop further iterations if replacement failed}
  end if
end for M_o, $\{(p_j, s_j, r_j)\}_{j=1}^{n}$ {Final molecule and list of edits}

The lists of predefined substituents (Table 3) and molecular linkers (Table 4) are provided for constructing synthetic editing samples.

A.2. Compound preprocessing and filtering

We extracted standardized SMILES strings and associated bioactivity data from the ChEMBL35 database. To ensure molecular quality and downstream compatibility, we applied the following filtering criteria:

- Salt removal: only the largest fragment was retained;
- Molecular weight between 100 and 800 Da;
- Allowed atom types: H, C, N, O, F, Cl, Br, I, S, P, B, Se;
- No linear unbranched chains longer than six heavy atoms.

These filters resulted in a curated set of drug-like molecules, which served as the input for matched molecular pair (MMP) generation.

Table 3. Common substituents for synthetic data.

Category	Name	SMILES
Halogens	Fluoro	[*:1]F
	Chloro	[*:1]Cl
	Bromo	[*:1]Br
	Iodo	[*:1]I
Alkyl	Methyl	[*:1]C
	Ethyl	[*:1]CC
	Isopropyl	[*:1]C(C)C
	tert-Butyl	[*:1]C(C)(C)C
Aryl	Phenyl	[*:1]c1ccccc1
	p-Tolyl	[*:1]c1ccc(cc1)C
	p-Chlorophenyl	[*:1]c1ccc(cc1)Cl
Oxygen-containing	Hydroxyl	[*:1]O
	Methoxy	[*:1]OC
	Ethoxy	[*:1]OCC
	Carboxyl	[*:1]C(=O)O
	Aldehyde	[*:1]C=O
	Ketone	[*:1]C(=O)C
Nitrogen-containing	Amino	[*:1]N
	Methylamino	[*:1]NC
	Dimethylamino	[*:1]N(C)C
	Cyano	[*:1]C#N
	Nitro	[*:1][N+](=O)[O-]
Sulfur-containing	Thiol	[*:1]S
	Methylthio	[*:1]SC
	Sulfonyl	[*:1]S(=O)(=O)C

A.3. Target grouping and MMP Analysis

After filtering, compounds were grouped by target ID, and each group was exported as a separate `.smi` file. For each group, matched molecular pairs (MMPs) were generated using the RDKit Contrib MMPA pipeline (`rfrag.py` and `indexing.py`). This procedure applies systematic bond fragmentation and indexing to identify molecular pairs that differ by a single localized transformation. Each transformation was encoded as a SMIRKS pattern, representing the minimal structural change. All MMP results were merged and formed a high-quality pool of target-specific matched molecular pairs.

A.4. Transformation type classification

To better characterize the nature of molecular edits, the resulting MMPs were further categorized into two types based on structural properties:

1. Core replacement. MMPs were classified as core replacements if they satisfied the following criteria:

- **Attachment point constraint:** both fragments have the same number of attachment points, with count ≥ 2.
- **Scaffold difference:** the Murcko scaffolds of the prior and latter fragments in replacement are different;
- **Ring requirement:** both fragments contain at least one ring;
- **Non-ring atom constraint:** each fragment contains no more than 5 non-ring heavy atoms, including the attachment points, to avoid heavily decorated rings;

Table 4. Common molecular linkers for synthetic data.

Category	Name	SMILES
Aromatic linkers	Meta-phenylene	[*:1]c1cc([*:2])ccc1
	Para-phenylene	[*:1]c1ccc([*:2])cc1
	Ortho-phenylene	[*:1]c1c([*:2])cccc1
Carbonyl-based linkers	Amide	[*:1][C;!R](=O)[N;!R][*:2]
	Reverse amide	[*:1][N;!R][C;!R](=O)[*:2]
	Ester	[*:1][C;!R](=O)[O;!R][*:2]
	Ketone bridge	[*:1][C;!R](=O)[*:2]
	Urea	[*:1][N;!R][C;!R](=O)[N;!R][*:2]
	Carbamate	[*:1][O;!R][C;!R](=O)[N;!R][*:2]
	Sulfonamide	[*:1]S(=O)(=O)[N;!R][*:2]
Alkyl / heteroatom linkers	Methylene	[*:1][C;!R][*:2]
	Ethylene	[*:1][C;!R][C;!R][*:2]
	Ether	[*:1][O;!R][*:2]
	Thioether	[*:1][S;!R][*:2]
	Secondary amine	[*:1][N;!R][*:2]
Extended / heterocyclic linkers	1,2,3-Triazole	[*:1]c1nnn([*:2])c1
	Imidazole-type	[*:1]c1[nH]cc([*:2])n1
	Piperazine	[*:1]N1CCN([*:2])CC1
	Piperidine	[*:1]N1CCC([*:2])CC1
Polar chain linker	PEG unit (ethylene glycol)	[*:1][O;!R][C;!R][C;!R][O;!R][*:2]

- **Component ratio:** the core fragment in the initial molecule account for less than 50% heavy atoms.

In cases where the original fragment exhibits symmetry and the replacement fragment does not (as illustrated in the last line of Figure 5), multiple distinct products may arise due to different permutations of attachment point mappings, accounting for 7.5% of our test set. Since the original attachment sites are chemically interchangeable, all such permutations are treated as valid ground truths for evaluation.

2. terminal replacement. MMPs were classified as single-point replacements (or terminal replacements) if the number of heavy atoms in the modified fragment accounts for no more than 30% of the initial molecule and only one attachment point.

A.5. t-SNE visualization of fragment chemical space

To assess the chemical diversity of the Realistic Edits test set relative to the synthetic moiety pool used in training, we computed ECFP fingerprints (radius = 2, 2048 bits) for all fragments and projected them into two dimensions using t-SNE. As shown in Figure 6, fragments from the synthetic training pool cover only a small portion of the chemical space and form several compact clusters, reflecting the limited structural motifs used to construct the synthetic edit pairs. In contrast, fragments from the Realistic Edits set are widely dispersed across the projection space, occupying diverse regions that are largely non-overlapping with the synthetic clusters.

This spread confirms that Realistic Edits cover a substantially broader and more diverse chemical space. Despite this distribution shift, the finetuned code LLM in MECo maintains strong performance, supporting our claim that the model learns generalizable code transformation patterns rather than relying on fragment-level memorization.

B. Model training details

B.1. Prompt format and design rationale

Prompt pre-filtering. To determine an optimal prompting strategy, we conducted preliminary experiments examining the effects of various formatting choices. These preliminary experiments were conducted under conditions that differ from our

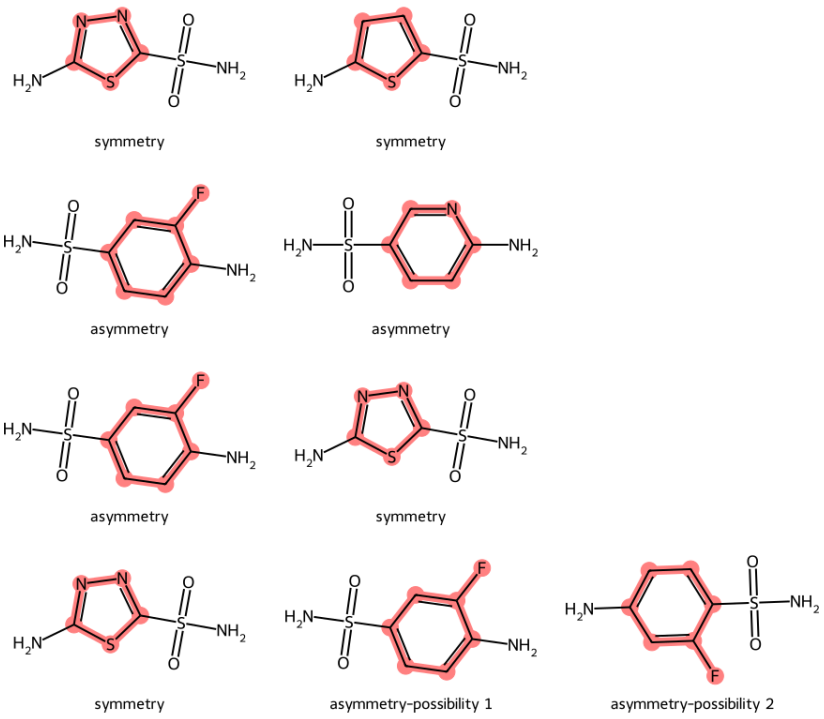

Figure 5. Multiple possible permutations in symmetry-to-asymmetry core replacement.

final evaluation tasks; they serve only as pilot studies to guide the choice of attachment-point style.

For numbering attachment points in the SMILES, we tested:

- Mark attachment points only in source SMILES with atom map numbers;
- Numerically number all atoms in SMILES and give the corresponding numbers of attachment points.

For marking attachment points in the fragment, we wrote original and target fragments with:

- atom map number, e.g., [C:1]N
- rooted at attachment point with a short dash, e.g., -CN
- a asterisk mark, or a so-called dummy atom, e.g., *CN

Table 5. Accuracy across different SMILES attachment styles and numbering strategies.

Action/Model	**Molecule Number Style**		**Attachment Point**	**Whole SMILES**
	Fragment Attachment Style			
CodeGen / Coder	Numbered	[C:1]N	0%	0%
	Rooted	-CN	20%	1%
	Asterisked	*CN	6%	6%
Direct / General	Numbered	[C:1]N	0%	3%
	Rooted	-CN	12%	6%
	Asterisked	*CN	4%	4%

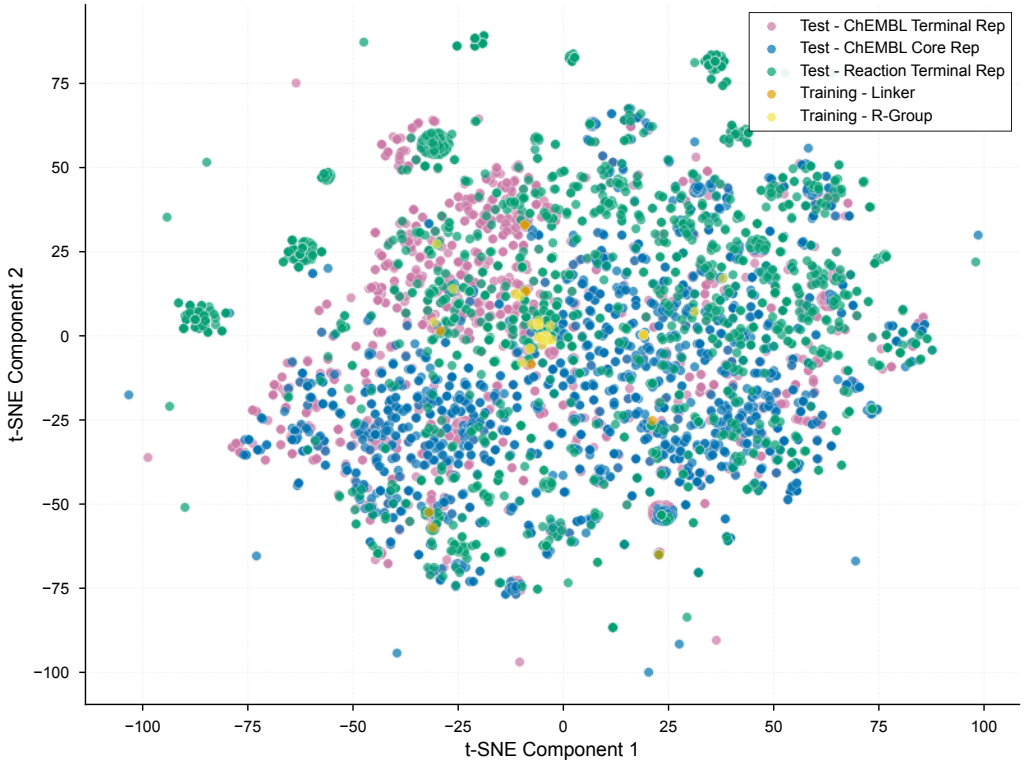

Figure 6. t-SNE for fragments in synthetic training data construction and realistic edit test set.

Preliminary results are shown in Table 5. While the asterisk symbol ($\star$) was not always the most accurate in pilot trials, it was retained because of its conventional role in representing dummy atoms in cheminformatics toolkits (e.g., RDKit, ChemDraw) and its compatibility with code generation. We also observed that passing numbered SMILES directly to the reasoning LLM leads to more stable responses, likely due to the limited power of LLMs handling SMILES during reasoning. In contrast, the rooted attachment-point style is not compatible with fragments requiring multiple attachment points, such as in core-replacement tasks.

Final Prompt Format

```
You are given a molecule in SMILES format:
"{numerically_numbered_source_smiles}".

For reference, atoms where new groups will be attached are marked with
"[*:n]", where "n" is the atom mapping number.

Future connected atoms in groups are labeled using the same numbers,
ensuring one-to-one attachment correspondence. You will then be given
multiple instructions on how to edit the molecule.

Replace the substructure corresponding to
"{original_fragment_smarts_1}"
connected at atom {numbers_indicating_attachment_points}
with "{target_fragment_smarts_1}".

...

Replace the substructure corresponding to
"{original_fragment_smarts_n}"
connected at atom {numbers_indicating_attachment_points}
with "{target_fragment_smarts_n}".

Generate a Python code snippet that performs these replacements
using RDKit via ChemicalReaction.

Ensure the code is executable and returns the modified molecule as
a new SMILES string.

You must return a Python code snippet wrapped in triple backticks.

The code should import modules from RDKit, perform the operation,
and print only the modified molecule in SMILES format.
```

B.2. Training setup

We used the official fine-tuning framework from the Qwen2.5-Coder repository[1] and conducted full-parameter supervised fine-tuning (SFT) using DeepSpeed on both `Qwen2.5-Coder-7B-Instruct` and `Qwen2.5-7B-Instruct` models. The training was performed on 4 NVIDIA A100 80GB GPUs, with each epoch taking approximately 4 hours.

C. Experiment details

C.1. Execution accuracy across fragment similarity bins

To further assess whether MECo generalizes beyond the structural motifs present in its synthetic training set, we evaluate execution accuracy across fragment similarity bins computed using both structural similarity (ECFP–Tanimoto) and string similarity (SequenceMatcher). Figure 7 reports accuracy curves over a wide range of similarity scores, covering multiple realistic edit test sources, including ChEMBL for activity-derived edits, USPTO for reaction-derived edits, and both terminal and core replacement types.

The results show that MECo maintains stable performance across the entire similarity spectrum, including bins where fragments are substantially dissimilar from those in the synthetic moiety pool. This provides further evidence that MECo's execution ability is not tied to neither structural coverage nor string pattern overlap of fragments in the training data, but instead generalizes robustly to structurally diverse and previously unseen edits.

[1] https://github.com/QwenLM/Qwen2.5-Coder/tree/main/finetuning/sft

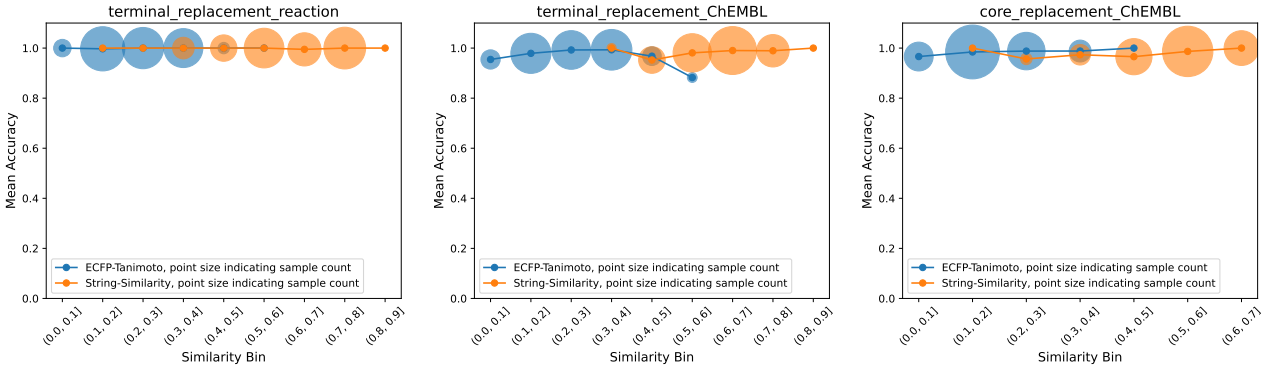

Figure 7. Execution accuracy of the finetuned code LLM (Qwen2.5-Coder-7B-Instruct-FT) across bins of structural similarity (ECFP–Tanimoto) and string similarity (SequenceMatcher), evaluated on data from different sources and for both terminal and core replacements. The size of the semi-transparent halo around each point reflects the number of test samples in that bin.

C.2. Prompt wrapper in the molecular optimization task

```
{Objective description}

Your response must be actions to perform the transformation, i.e.
"replace original_group_smiles connected at atom_number with
target_group_smiles".

Each action must include two SMILES strings indicating the original
group and the target group.

Use dummy atoms to mark the connection point, e.g.
"replace [*:1]Cl connected at atom_number with [*:1]OC"
or
"replace [*2:]c1ccc([*:3])cc1 connected at atom_number2, atom_number3
 with [*2:]c1cnc([*:3])cc1".

Your response must be in directly parsable JSON format:
{
    "Action Description": [
        "replace original_group_smiles connected

        at atom_number with target_group_smiles",

        ...
    ],
    "Final Target Molecule": "SMILES"
}

Given the source molecule with atoms numbered:
{number_smi(source_smiles)}.

You should ignore Hydrogen (H) and numbers in the group SMILES.
```

C.3. Criteria and evaluation for consistency manual check

Criteria. We considered syntax errors in actions that are still discernible to human experts. Examples include malformed atom map annotations such as `[*1:]` or `[*1]`, which should be `[*:1]`, and missing explicit hydrogens on aromatic nitrogens, e.g., `[*:1]c1nnnn1`, which should be `[*:1]c1[nH]nnn1`.

For attachment point checks, we adopted relatively loose criteria. For instance, in `[cH:1]1[cH:2][cH:3][cH:4][cH:5][c:6]1[O:7][CH2:8][CH:9]1[CH2:10][CH2:11]1`, assign-

ing the attachment of `[*:1]OCC1CC1` to atom 6 or 7 is both considered discernible. Similarly, in `[*:1]O[*:2]`, providing only one side of the atom index (e.g., 6 or 8) or the atom index alone (7) is also regarded as discernible. By contrast, edits that invoke groups not present in the structure are considered invalid, even if similar patterns can be found.

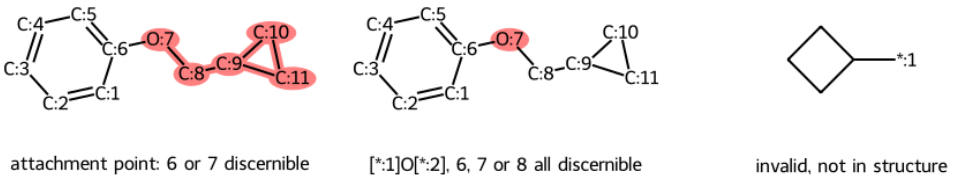

Figure 8. Visualization of manual check criteria.

Nevertheless, manual checking is inherently subjective. Different researchers may interpret borderline cases differently, and there remains a possibility of human error.

Subjectivity. To further assess potential subjectivity, we adopt a blinded multi-annotator protocol: three annotators independently evaluate 120 samples (evenly drawn from DeepSeek-R1), without access to model identities, task provenance, or one another's labels. Structural labels (SMILES) were likewise highly consistent: three-way agreement = 95.83%, two-way = 3.33%, and complete disagreement = 0.83%. Attachment-point partiality judgments achieved Fleiss' $\kappa = 0.9342$ for the 104 jointly valid cases, indicating almost perfect agreement.

Scalability. This verification pipeline naturally supports scalable dataset growth: for each natural-language edit instruction, annotators provide one gold-standard structure (e.g., SMILES) that correctly implements the edit. Each labeled (instruction, targeted structure) pair becomes a reusable test case and can be automatically applied to any future code-LLM output via structure matching. Thus, each round of human labeling incrementally expands a standardized, reproducible, and low-cost evaluation suite. In addition, the annotation process can be substantially accelerated through formalized validation code. For example, automatically excluding substructure errors (e.g., fragments not present in the source molecule) before human inspection. This further reduces subjective load and ensures that annotators only review chemically valid candidate edits.

Regarding scalability toward larger and more diverse test sets, we draw on recent progress in LLM evaluation and outline several feasible extensions:

1. **LLM-as-a-Judge with minimal supervision.** Lin & Chen (2023); Li et al. (2025b) show that structured prompting enables LLMs to reliably approximate human judgments at scale, suggesting that MECo's intention-consistency checks could be extended through supervised automatic judges.

2. **Decomposed, verifiable evaluation dimensions.** Zhou et al. (2023); Ye et al. (2023) shows that breaking complex behaviors into verifiable atomic criteria yields more stable and interpretable assessments. Future MECo versions may evaluate edits along dimensions such as (i) substructure localization, (ii) bond/valence correctness, (iii) atom-mapping agreement, and (iv) chemically valid topology, paired with automatic graph/SMILES validators. This reduces subjectivity and aligns evaluation with chemically interpretable components.

3. **Standardized benchmark construction.** Successful evaluation suites (Zhou et al., 2023; Zheng et al., 2023) emphasize fixed task sets, unified templates, and executable scoring pipelines. Inspired by these designs, we could build a standardized benchmark for molecular edit–intention consistency with public task suites and unified evaluation scripts, ensuring fully reproducible cross-model comparison.

These extensions provide a clear and technically grounded path toward scaling MECo's intention-consistency evaluation beyond the manually assessed subset while enhancing reproducibility, transparency, and objectivity.

C.4. Observation on terminal replacement test samples

In Section 4.2, we noted a discrepancy between reaction-derived and target-specific terminal replacements. To investigate this further, we conducted a closer inspection and found that the discrepancy may stem from differences in SMILES syntax patterns: in target-specific edits, dummy atoms (e.g., `[*:1]`) often appear in the middle or at the end of SMILES strings,

whereas in reaction-derived edits, they frequently occur at the beginning. This difference arise from the implementation of the extraction program rather than the underlying data distribution, meaning that the two syntax types can be transformed into each other by choosing whether the dummy atom or another atom is used as the SMILES root. Such pattern sensitivity suggests that LLMs struggle to generalize over SMILES syntax, further motivating the use of structured code as a more robust and interpretable intermediate representation.

C.5. Property improvement vs. similarity

To further analyze the trade-off between property improvement and structural preservation, we compare the results of direct generation and MECo separately on all six tasks (Figure 9). The top row shows the average property improvement (relative to the initial molecule), where MECo generally achieves higher improvements than direct generation. The bottom row shows the average structural similarity (ECFP4-based) to the initial molecule. While Gemini-2.5-Pro exhibits larger improvements in the upper row, it tends to reduce similarity more than DeepSeek-R1, indicating that different LLMs show different levels of willingness to sacrifice structural similarity in exchange for property gains. This observation motivates our use of success rate as the primary evaluation metric in the main paper, as it balances both optimization effectiveness and structural plausibility.

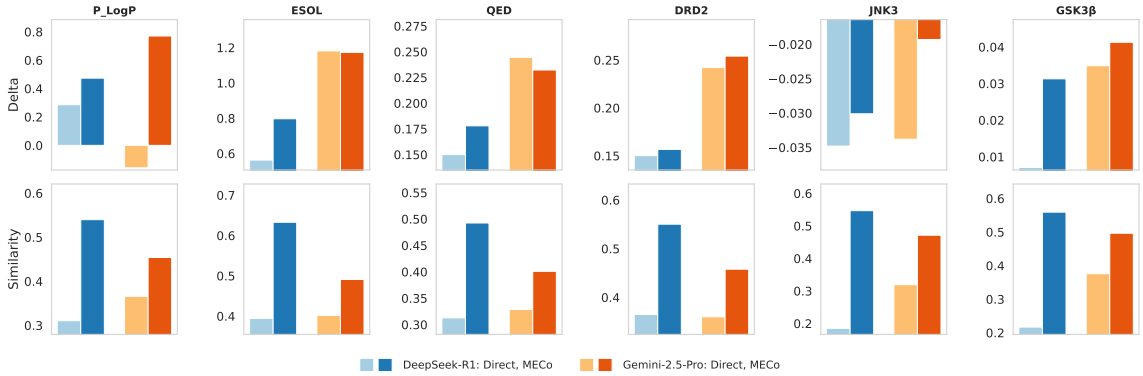

Figure 9. Bar plots of molecular optimization performance across six tasks, showing property improvement (top row) and structural similarity (bottom row) for Direct generation and MECo for DeepSeek-R1 and Gemini-2.5-Pro.

C.6. Docking setups for case study

For the case study, molecular docking was performed using the Glide SP (standard precision) protocol implemented in the Schrödinger suite. The high-resolution X-ray co-crystallized target protein structure was obtained from the Protein Data Bank (PDB ID: 4AFJ).

Protein preparation was carried out using the Protein Preparation Wizard, including addition of hydrogen atoms, assignment of bond orders, optimization of the hydrogen-bonding network, and restrained minimization of heavy atoms with the OPLS_2005 force field. Ligands were prepared using LigPrep to generate at most 32 stereoisomers.

Docking was performed with default Glide SP parameters unless otherwise specified. The receptor grid was centered on the co-crystallized ligand SJJ in 4AFJ. The top-ranked docking poses were retained for analysis and visualization.