MOLLANGDATA: A LARGE-SCALE DATASET FOR MOLECULAR STRUCTURE–LANGUAGE DESCRIPTION VIA A RULE-REGULARIZED METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular function is largely determined by structure. Accurately aligning molecular structure with natural language is therefore essential for enabling large language models (LLMs) to both understand structure for chemical reasoning and generate molecules from natural-language design intent. However, the substantial cost of human annotation makes it infeasible to construct large-scale, high-quality datasets of structure-grounded descriptions. This work proposes a fully automated annotation framework for generating precise molecular descriptions at scale, such that the original molecule can be unambiguously reconstructed from the description alone. Our approach extends a rule-based chemical nomenclature parser to interpret IUPAC names and construct enriched, XML metadata that explicitly encodes molecular structure. This is then used to guide LLMs in producing accurate natural-language descriptions. Using this framework, we curate MolLangData, a dataset of approximately 163k molecule–description pairs. A rigorous validation protocol combining expert human and LLM-based reconstruction on a subset of 2,000 molecules demonstrates 98.6% description precision. Using the curated dataset, we train a 4B-parameter LLM via large-scale reinforcement learning for language-conditional molecule generation. Despite its much smaller size, the resulting model significantly outperforms most of the frontier LLMs while using far fewer tokens. The proposed framework and dataset provide a reliable foundation for molecule–language alignment, readily beneficial to broader chemical tasks.

1 INTRODUCTION

By aligning other modalities with language, the strong reasoning capabilities, flexibility, and extensive knowledge embedded in large language models (LLMs) can be transferred to multimodal settings, allowing for interpretation and inference beyond text. This paradigm has proven particularly successful in vision–language models (VLMs). For example, VLMs are now widely used for image recognition and analysis (OpenAI, 2026a), as well as for conditional image generation and editing based on user instructions (OpenAI, 2026b). Similarly, aligning molecular representations with language could enable chemical reasoning that is currently not possible for traditional unimodal models (Cai et al., 2025; Irwin et al., 2022), thereby providing new opportunities for diverse chemical tasks such as property prediction (Wu et al., 2018), molecular and drug discovery (Brown et al., 2019; Polykovskiy et al., 2020), and synthesis prediction and planning (Lowe, 2017).

Early attempts to bridge molecular representations with language include the curation of molecule-description datasets (Edwards et al., 2021; 2024) and subsequent models that translate between the two modalities (Edwards et al., 2021; 2022). With the emergence of LLMs, research efforts have shifted toward leveraging them for downstream chemical tasks, including direct capability benchmarking (Guo et al., 2023), training chemistry- or science-specific LLMs (Taylor et al., 2022; Zhang et al., 2024), and building multimodal models (Su et al., 2022; Jablonka et al., 2024; Liu et al., 2025). However, these approaches remain far from practical use and still lag behind unimodal chemical models on nearly all downstream tasks. More detailed related work is discussed in Appendix A.

These models, along with curated datasets, bypass alignment between language and the underlying molecular structure. Instead, they attempt to directly bridge the molecular representations with

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.

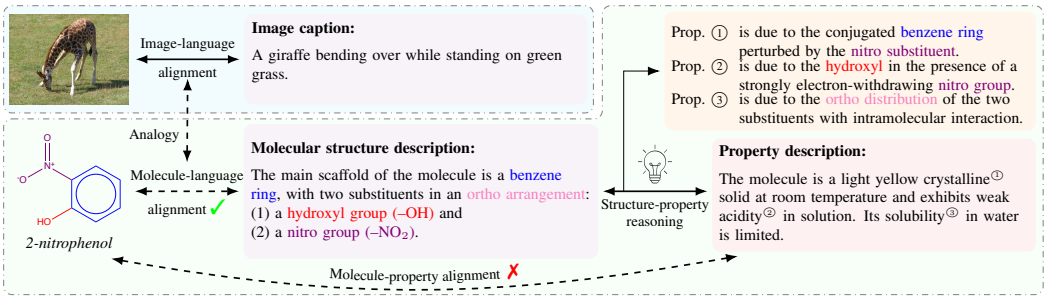


Figure 1: An illustrative example motivating this work. Existing approaches align molecular representations with high-level objectives; we argue molecule–language alignment should be structure-grounded, with higher-level reasoning left to the LLM backbone, analogous to image–language alignment. Real molecular descriptions are far more complex; see Appendix Fig. B.1 and Sec. E.3.

downstream objectives that require higher-level, domain-specific reasoning. However, as illustrated in Fig. 1, a molecule's function, including physicochemical properties and reaction mechanisms, is determined by its structure. Whether performing property inference or molecular design, chemists always rely on an explicit understanding of molecular structure, and so should AI models. This gap has been systematically highlighted by MolLangBench (Cai et al., 2026), which evaluates molecule–language interface tasks such as molecular structure recognition, and structure generation and editing from language. The results show that even advanced general-purpose LLMs perform far from perfectly on these tasks, while chemical-specific multimodal models fail almost entirely. Without these foundational capabilities, success on more complex chemical reasoning tasks is unlikely.

We therefore argue that *molecular representations should first be aligned with structure-grounded language descriptions*, with chemical knowledge and functional reasoning then encoded within the language model through large-scale training. This resonates with vision–language modeling: image descriptions focus on faithfully describing the *observed visual content*, as illustrated in Fig. 1, with the visual module serving primarily as a perception or generation component, while deeper interpretation and reasoning are delegated to the LLM backbone (Zhou et al., 2025).

Molecule–language alignment thus demands extensive structure-grounded descriptions, as in VLMs (Schuhmann et al., 2022). However, curating molecular description datasets presents different challenges. First, subtle structural changes can dramatically alter properties and reaction mechanisms (Eichelbaum et al., 2012; Smith, 2020), requiring descriptions to be *complete and unambiguous*. Second, unlike the image domain, no large-scale molecular structure descriptions readily exist. Given the complexity of molecular structures, including intricate ring topologies, nested substituents, and stereochemical configurations, human annotation at scale is also impractical.

We address this challenge by developing an automated annotation framework. Building upon a rule-based IUPAC name parser, OPSIN (Lowe et al., 2011), we invest considerable engineering to construct enriched XML metadata encoding molecular structure. This metadata is injected into LLMs to guide generation of descriptions that preserve complete structural details, including ring topology, substituent positioning, attachment relations, labeling semantics, and stereochemical information. Using this pipeline, we curate MolLangData, a dataset of $163{,}111$ molecule–description pairs. Rigorous validation on a $2{,}000$-sample subset by reconstructing the exact structures from descriptions, involving substantial expert effort, demonstrates 98.6% annotation precision. Then, we train a 4B-parameter LLM on MolLangData via large-scale reinforcement learning for language-conditional molecule generation. The training improves the base model's generation accuracy from 12.53% to 84.99%. Though much smaller, our model significantly outperforms most evaluated frontier LLMs. Meanwhile, the model shows strong token efficiency, reducing token usage by $10\times$ compared with open-weight LLMs. Together, the quality and utility of MolLangData are well justified, establishing a reliable foundation for molecule–language alignment toward advancing chemical reasoning.

2 Task Formulation and Challenges

We consider the task of generating a large-scale dataset of molecular structure descriptions. Specifically, each entry consists of a molecular structure paired with a corresponding structural description,

denoted $(\mathcal{M}, \mathcal{T})$. The description $\mathcal{T}$ is *complete and unambiguous* such that the molecule $\mathcal{M}$ can be uniquely determined from it. Conversely, a given molecular structure may admit multiple valid descriptions due to linguistic and chemical terminology variation, but all should be semantically equivalent. Generating such a dataset at scale presents several challenges:

Challenge 1: Appropriate Level of Structural Detail At one extreme, a description could enumerate all atoms, bond types, connectivities, and stereochemical relationships. Although such descriptions can be generated by traversing the molecular graph, they become verbose even for molecules of moderate size. More importantly, molecular properties and chemical reactions are associated with functional groups and scaffold-level motifs rather than isolated atoms. Purely atom-level descriptions therefore fail to align structure with the abstractions most useful for downstream reasoning. At the other extreme, descriptions that only summarize functional groups or scaffold types may omit critical structural details. Subtle differences in ring topologies, attachment positions, or stereochemistry can substantially change physicochemical properties, biological activity, toxicity, or reaction mechanisms (Eichelbaum et al., 2012; Smith, 2020), and cannot be ignored. An effective description must therefore balance sufficient structural specificity with chemically meaningful abstractions.

Challenge 2: Scalability of Dataset Construction Molecule–language modeling requires datasets containing hundreds of thousands to millions of molecule–description pairs. According to Cai et al. (2026), annotating and validating a single structural description can require up to $1\,\mathrm{h}$ of expert effort. Even hybrid pipelines combining LLM-assisted generation with human validation remain impractical at this scale. Dataset construction must therefore rely on fully automated generation.

Challenge 3: Insufficient Structural Information LLMs are increasingly used to scale annotation (Tan et al., 2024), but generating accurate structural descriptions depends on complete and precise structural information, which is not conveyed by common molecular representations. Linear representations such as SMILES (Weininger, 1988) specify atom-wise connectivity through grammar rules and traversal order, but do not encode structural abstractions. LLMs must instead infer abstractions from atom-level sequences, which we and Cai et al. (2026) find to be error-prone, particularly for complex ring topologies and long side chains. In contrast, IUPAC nomenclature more closely reflects how chemists describe molecular structures. However, correct interpretation of IUPAC names requires extensive nomenclature knowledge, including tokenization conventions, locant assignment, ring labeling schemes, and fusion, bridged, and spiro descriptors. As illustrated in Fig. 2, interpreting the name requires understanding how to tokenize, where and in what order to parse, what fusion letters mean (e.g., the 'e' in fused-ring notation), and how to assign locants after complex ring fusion, bridging, or spiro structures. The IUPAC Blue Book (Favre & Powell, 2013) takes over $1{,}100$ pages on organic compound nomenclature. Consequently, generating descriptions directly from IUPAC names remains unreliable, as demonstrated in Appendix Ablation Study B.4.

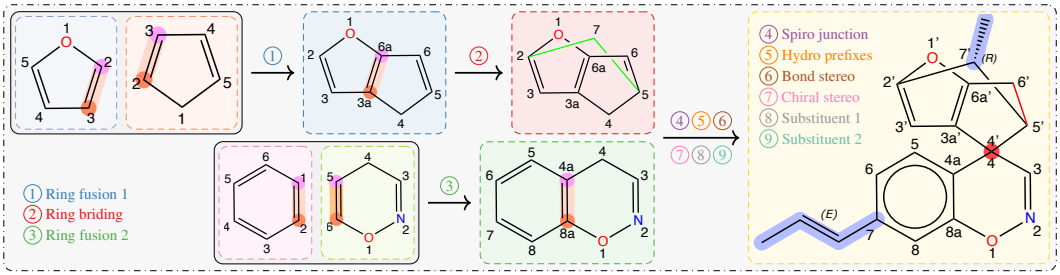
3 Methodology of Dataset Generation

Building on OPSIN (Lowe et al., 2011), a rule-based system for interpreting chemical nomenclature and reconstructing structures, we develop an automated pipeline that substantially adapts and extends its intermediate representations to produce XML metadata encoding molecular structure. This metadata is used to guide LLMs in generating complete and unambiguous structure descriptions. Below, we first briefly introduce OPSIN and analyze the limitations of its native parse tree for description generation, then present our approach for constructing enriched structural metadata, and finally describe the prompt design and filtering strategies used to improve generation accuracy.

3.1 Limitations of OPSIN for Description Generation

Given a chemical name, OPSIN performs grammatical tokenization (e.g., *propan-2-ol* $\rightarrow$ [*prop, an, -, 2-, ol*]) and assigns each token a semantic role (e.g., root structures, substituents, locants, suffixes, stereochemistry). The tokens and their attributes are organized into an XML parse tree representing intermediate structural interpretations. The tree is then operated through a staged pipeline that generates, processes, and assembles structural components into a molecular graph, output as CML (Murray-Rust & Rzepa, 1999), InChI (Heller et al., 2013), or SMILES (Weininger, 1988).

However, OPSIN's XML is designed as an internal representation rather than a complete, accurate structural encoding, making it unsuitable for guiding description generation. Specifically, (i) struc-




```
<molecule name="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]oxazine-4,4'-[2,5]methano
↪  cyclopenta[b]furan]">
 <wordRule wordRule="simple" type="full"
 ↪  value="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]ox...">
  <word type="full" value="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]ox...">
   <substituent locant="7'">
    <group type="chain" subType="alkaneStem" value="C" labels="numeric" usableAsAJoiner="yes"
    ↪  resolved="yes">meth</group>
    <suffix type="inline" value="yl">yl</suffix>                                                    ⑨
    <hyphen value="-">-</hyphen>
   </substituent>
   <bracket locant="7">
    <substituent>
     <stereoChemistry type="EorZ" value="E">E</stereoChemistry>                                     ⑥
     <group type="chain" subType="alkaneStem" value="CCC" labels="numeric">
      <unsaturator value="2" subsequentUnsemanticToken="-" locant="1">en</unsaturator>              ⑧
     </group>
     <suffix type="inline" value="yl" locant="1">yl</suffix>
    </substituent>
    <hyphen value="-">-</hyphen>
   </bracket>
   <root>
    <stereoChemistry locant="7'" type="RorS" value="R" stereoGroup="Abs">7'R</stereoChemistry>      ⑦
    <group type="spiro system" subType="Non-Identical Polycyclic" value="spiro, benzo[e]oxazin,
    ↪  methpent[b]furan">
     <spiroSystemComponent type="ring" subType="fusedRing" value="benzo[e]oxazin">
      <fusedChildRing type="ring" subType="hantzschWidman" value="c1cccccl" labels="1/2,ortho/3,meta/4,para/5/6">
       <heteroatom value="O" locant="1" resolved="yes">ox</heteroatom>
       <heteroatom value="N" locant="2" resolved="yes">az</heteroatom>
      </fusedChildRing>
      <fusedChildRing type="ring" subType="fusionRing" value="c1cccccl"                             ③
      ↪  labels="1/2/3/4/5/6">benzo</fusedChildRing>
      <fusedRingLabels labels="1/2/3/4/4a/5/6/7/8/8a"
      ↪  originalLabels="(1,)/(2,)/(3,)/(4,)/(5,1)/(,6)/(,5)/(,4)/(,3)/(6,2)"></fusedRingLabels>
     </spiroSystemComponent>
     <spiroLocant>4,4'</spiroLocant>                                                                ④
     <spiroSystemComponent type="ring" subType="bridgeSystem" value="methpent[b]furan">
      <bridgeParent type="ring" subType="fusedRing" value="pent[b]furan">
       <fusedChildRing type="ring" subType="ring" value="olcccl" labels="1/2/3/4/5">furan</fusedChildRing>
       <fusedChildRing type="ring" subType="alkaneStem" value="ClCCCCl" labels="numeric"
       ↪  conjugated="true">pent</fusedChildRing>                                                   ①
       <fusedRingLabels labels="1/2/3/3a/4/5/6/6a"
       ↪  originalLabels="(1,)/(5,)/(4,)/(3,2)/(,3)/(,4)/(,5)/(2,1)"></fusedRingLabels>
      </bridgeParent>
      <bridgeChild type="chain" subType="alkaneStem" value="-C-" labels="7"
      ↪  bridgeLocants="2,5">meth</bridgeChild>                                                     ②
     </spiroSystemComponent>
     <hydro value="hydro" multiplied="multiplied" locant="6'">hydro</hydro>
     <hydro value="hydro" multiplied="multiplied" locant="5'">hydro</hydro>                         ⑤
    </group>
   </root>
  </word>
 </wordRule>
</molecule>
```

Figure 2: Decomposition of an example molecule from basic components to the complete structure, along with the XML metadata constructed by our approach. The native OPSIN XML output and generated natural-language description are provided in Appendix Figs. B.2 and B.1, respectively.

tural elements such as substituents, locants, suffixes, and stereochemical descriptors are often not positioned to reflect their true attachment or affiliation in the final structure; (ii) many intermediate elements are discarded once used, leaving the final XML representation incomplete; (iii) topological relationships, such as fused-ring connectivity, bridged and spiro junctions, implicit locants, and atom labeling schemes, are handled internally and never recorded; and (iv) some components (e.g., von Baeyer spiro systems) are resolved directly into SMILES, from which alone LLMs struggle to interpret structure. Enriched structural metadata tailored for description generation is thus needed.

3.2 Enriched Structural Metadata Construction

We fully leverage OPSIN's grammatical tokenization and initial XML parse tree construction. We perform substantial engineering to transform the parse tree into a metadata representation suitable

for guiding molecular structure description generation. Importantly, these modifications do not alter OPSIN's native structure construction process. Our tool serves as an adaptive extension of OPSIN, preserving full compatibility with chemical name-to-structure conversion while providing a semantically complete and well-organized structural representation. This design facilitates compatibility with future OPSIN updates and a principled sanity check by comparing converted structures with those from the native OPSIN pipeline, ensuring the reliability of generated metadata.

A systematic redesign of OPSIN is impractical, as both IUPAC nomenclature and OPSIN are highly complex; despite being actively developed and maintained for over a decade, OPSIN continues to encounter unresolved special cases. We therefore adopt a trial-and-error engineering strategy. We iteratively test and refine our metadata construction on over $2{,}000$ molecules covering diverse and challenging cases, including but not limited to, complex fused, bridged, and spiro ring systems; Hantzsch-Widman heterocycles; multiple stereochemical configurations; amino acid derivatives; and a wide range of prefixes, infixes, and suffixes. Using the representative example in Fig. 2 and the corresponding native OPSIN XML in Appendix Fig. B.2, we summarize the key modifications:

Augment topology and labeling information. Molecules containing complex ring systems comprise over 30% of entries in public repositories such as PubChem (Kim et al., 2016). In native OPSIN, ring-construction information and the resulting atom labeling schemes are processed internally but not written in the parse tree. This missing information is critical both for accurately describing ring topology and correctly determining the attachment locants for subsequent structural components. We therefore augment the XML to encode the missing topology and atom labeling information.

For fused rings, we first record each base ring as a `fusedChildRing` element, storing its atom set (`value` attribute) and original atom labels (`labels` attribute). A `fusedRingLabels` structure then specifies both the fusion junctions between the base rings (`originalLabels` attribute) and the new atom labeling scheme of the fused system (`labels` attribute). In the *benzo[e][1,2]oxazine* fusion (process ③), *oxazine* and the *benzo* rings are fused by sharing two atom pairs, $(5, 1)$ and $(6, 2)$: *oxazine* atoms 5 and 6 are fused with *benzo* atoms 1 and 2, respectively. After fusion, these junction atoms are relabeled as $4a$ and $8a$. The complete fusion process is captured in the augmented metadata. Bridged and spiro ring systems are handled analogously. These representations support cascaded multi-ring fusions and nested topologies, as in the illustrative example. Semantic definitions for fused, bridged, and spiro systems are provided in Prompts 4, 5, and 6 in the Appendix E.2.

Retain critical elements. We modify OPSIN to retain structurally meaningful elements that are otherwise discarded during component generation and structure assembly. As shown in Fig. 2, stereochemical information (⑥ and ⑦), unsaturation descriptors such as the *en* modifier of the *prop* substituent (⑧), hydro prefixes (⑤), and core ring descriptors, including heteroatom information in the *oxazine* ring (③), are removed in the native OPSIN parse tree but explicitly retained in our metadata. In addition, fused, bridged, and spiro ring information, which native OPSIN never creates and reduces to a generic systematic scaffold name, is fully augmented in our XML representation (①–④), as described above. All retained elements are further marked and post-processed to prevent re-interpretation or unintended modification in subsequent OPSIN processing stages.

Rearrange elements to reflect affiliation and connectivity. The native OPSIN XML organizes elements according to IUPAC token order, whereas actual modification and attachment relationships are resolved through rule-based inference during structure assembly. Thus, locants, prefixes, and stereochemical descriptors may appear detached from the structural components they actually modify. For example, in *(E)-5-(prop-1-en-1-yl)non-3-ene*, the '*E*' descriptor applies to the *non-3-ene* backbone, whereas in *(E)-5-(prop-1-en-1-yl)non-1-ene*, it instead applies to the *prop-1-en-1-yl* substituent. A similar issue arises in the illustrative example, where prefixes such as hydrogenation modifiers should apply to the fully resolved fused or spiro scaffold rather than to early name fragments. However, the native OPSIN parse tree does not encode these relationships. We systematically relocate prefixes, locants, stereochemical descriptors, and substituents in the XML tree so that each element is attached to the structural component it modifies in the final molecular graph.

Miscellaneous corner cases. In addition to the major structural deficiencies discussed above, our practical implementation required substantial engineering effort to address many additional corner cases where native OPSIN metadata is incomplete or difficult for LLMs to interpret. For example, explicit heteroatom positions in Hantzsch-Widman systems (e.g., *1,2-oxazole*) are not encoded in the native XML representation. Similarly, for von Baeyer spiro systems, OPSIN directly resolves name

fragments into SMILES strings via rule-based functions, which are effective for structure construction but provide limited and unreliable guidance for LLM-based description generation. Although corner cases remain, our iterative refinement enables construction of complete and structured metadata for the vast majority of molecules encountered in practice, as supported by our validation results showing that no description errors result from incomplete or incorrect metadata.

3.3 Prompt Design and Atom-Matching Filtering

In addition to our constructed metadata, the LLM is provided with the corresponding SMILES and IUPAC name, and is instructed to generate a self-contained, natural-language structure description. The prompt used for description generation is given in Appendix Prompt 2. The generated description is required to be sufficiently complete and precise that someone with basic organic chemistry knowledge can reconstruct the structure exactly and unambiguously from the description alone. The prompt includes guidance on describing molecular backbones, connectivity, substituents, functional groups, and stereochemistry. When fused, spiro, or bridged ring systems are present, additional semantic specifications are injected to guide the interpretation of complex ring topologies.

Despite the detailed structural information provided, preliminary experiments show that the LLM occasionally omits atoms in long side chains or chain linkers. To mitigate this, we incorporate an automated atom-matching filtering step during description generation. In addition to description generation, the LLM is prompted to count the total non-hydrogen atoms based solely on its generated description. Descriptions whose reported atom counts do not match the ground truth are discarded.

More rigorous self-checking mechanisms are possible, such as prompting the LLM to reconstruct the SMILES from the generated description and performing full structure matching. However, this strategy is not suited for large-scale dataset construction. First, including SMILES reconstruction within the same prompt introduces a non-trivial task that can interfere with the primary objective of structural description generation; alternatively, performing reconstruction in a separate LLM call substantially increases cost, making it impractical at scale. Second, single-pass reconstruction checking suffers from high false rejection rates: even strong models can reject a large fraction of otherwise correct descriptions, resulting in low annotation efficiency despite near-zero description errors. We therefore adopt this lightweight yet effective atom-matching filter, which improves description accuracy while preserving scalability, as demonstrated in the ablation study (Sec. B.4 in the Appendix).

4 Dataset Collection and Validation

4.1 Dataset Collection

Chemical space is extremely large (Reymond, 2015); even public molecule repositories contain hundreds of millions to billions of molecules (Kim et al., 2016; Irwin et al., 2020). We randomly sample $200{,}000$ molecules from PubChem (Kim et al., 2016) as candidates for dataset construction. We then apply initial data-quality and consistency checks before description generation. After filtering, $167{,}416$ molecules remain and are passed to the LLM for description generation. Multi-component entries account for 88.7% of excluded candidates. OPSIN and our tool natively support these entries, and each component can also be processed separately. We exclude them only to simplify validation. The remaining exclusions are mainly due to missing IUPAC names or mismatches caused by errors in the PubChem SMILES records. Fewer than 1% of candidates are excluded due to OPSIN parsing limitations. Detailed filtering statistics are provided in Appendix B.2.

While structural metadata provides sufficient information, accurate generation still requires the LLM to faithfully translate it into precise textual descriptions. We therefore use GPT-5.2 (OpenAI, 2025b) as the generation model (the most capable model available at the time of dataset curation). We group molecules into three generation difficulty levels, and use high reasoning effort for easy molecules and xhigh for medium and hard molecules. The difficulty criteria and the model snapshots used for generation are provided in Appendix B.3 and B.5, respectively. Note that *the difficulty level does not necessarily reflect molecular complexity*; molecules in the easy category may still be structurally intricate, as illustrated in Appendix Fig. E.1b. As reported in Appendix B.9, curating the dataset with GPT-5.2 incurred over \$20k in API costs. While higher-capacity models like GPT-5.5 can produce more accurate descriptions, they cost $2\times$ as much. The choice of generation model and reasoning effort for each difficulty level balances generation quality and practical budget constraints.

We additionally prompt the LLM to count non-hydrogen atoms to filter potentially erroneous descriptions. After filtering, the final dataset consists of 163,111 molecule–description pairs, with the distribution across difficulty levels summarized in Table 1. On average, each molecule contains 26.24 heavy atoms, and its corresponding description consists of 261 words. Representative examples and additional dataset statistics are provided in Appendix E.3 and B.8, respectively.

4.2 Dataset Validation

To assess reliability, we validate a random subset of 2,000 from the dataset that pass the atom-matching check, comprising 1,317, 496, and 187 easy, medium, and hard samples, respectively.

Validation Process Human validation of molecular descriptions is time-consuming; based on our records, validating a single sample requires approximately 11.7 min on average, slightly above the 10 min reported by Cai et al. (2026). Fully validating even the 2,000-sample subset thus requires about 390 human-hours for a single round, excluding additional validation for ablation studies.

To evaluate more samples while keeping human validation practical, we combine LLM-based and human validation. For each sample, we prompt GPT-5.2, GLM-5.1, and DeepSeek-V4-Pro to reconstruct the molecule in SMILES solely from its description (Appendix Prompt 1), with up to three independent attempts per model. A sample is accepted only when *all* three LLMs exactly reconstruct the ground-truth structure. We expect near-zero false positives because exact reconstruction from an incorrect or ambiguous description by chance is unlikely. Even for the 626 samples accepted by only one or two LLMs and thus rejected by automatic validation, later human validation confirms unambiguous descriptions and exact reconstruction. Requiring agreement among all three LLMs provides an additional safeguard. Human validation of 200 randomly selected samples accepted by all three also finds no false positives. Appendix B.7 provides the full empirical analysis. The LLMs successfully validate 62.8% of samples, reducing the number requiring human validation to 744.

However, LLM validation may produce false negatives. All samples failing LLM validation undergo human validation, where each sample is independently evaluated by up to two expert validators. Validators are provided only with the textual description and reconstruct the molecule using chemical drawing software; their reconstructed structures are submitted to an internal validation application that automatically checks them against the ground truth. Each validator is allowed up to three attempts. A sample is considered valid if either expert confirms that the description is *unambiguous* and that they successfully reconstruct the *exact* structure. A sample is labeled incorrect only if both validators fail to reconstruct it, mitigating the risk of individual oversight. Although LLM validation reduces the human workload, expert validation still requires approximately 160 h. We additionally spend around 110 h on validation for the false-positive study and the ablation studies described later.

Validation Results We report description precision on the validation set across difficulty levels in Table 1, with additional validation statistics in Appendix B.6. The overall precision of validated samples is 98.6%.

Table 1: Dataset statistics across generation difficulty levels, including collected samples, validation subset size, and validation precision (%). For collected and validation samples, each entry is reported as the sample count with its proportion (%) relative to the total dataset or total validation set, respectively.

Difficulty	Collected samples	Validated samples	Correct samples
Easy	106,402 (65.2%)	1,317 (65.8%)	1,300 (98.7%)
Medium	41,412 (25.4%)	496 (24.8%)	492 (99.2%)
Hard	15,297 (9.4%)	187 (9.4%)	180 (96.3%)
Overall	**163,111**	**2,000**	**1,972** (98.6%)

We examine all incorrect descriptions and confirm that structure parsing and metadata construction are correct; errors therefore arise solely from LLM description generation. In all incorrect cases, the errors are minor: descriptions contain all scaffolds and substituents, with most structural elements correctly connected and only localized issues, such as misplaced substituents or imprecise ring fusion descriptions. Overall, the low error rate and minor errors demonstrate the reliability of the generation pipeline and the resulting dataset.

4.3 Ablation Study

We perform ablation studies to analyze the impact of our constructed molecular-structure XML metadata, atom-matching filtering, and the choice of generation models. Each component meaningfully affects description precision. Full results and analysis are provided in Appendix B.4.

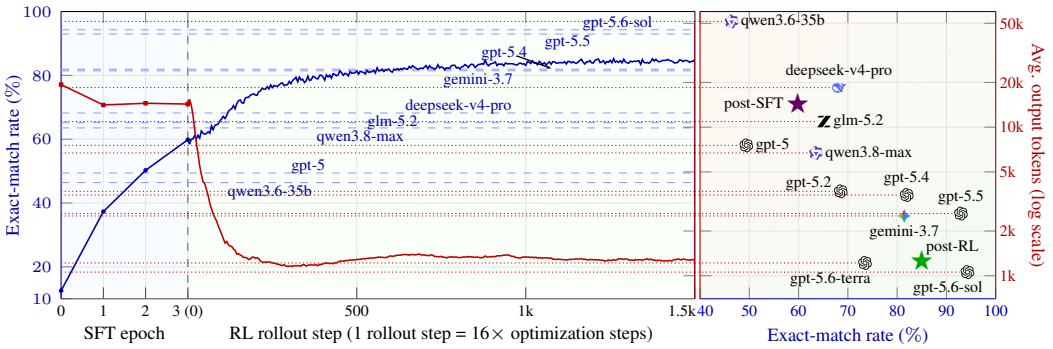


Figure 3: The left panel shows the training trajectory of exact-match rate and average output tokens on our test set during SFT and RL, with horizontal reference lines for selected LLM baselines. The right panel compares the final post-RL model with these baselines. Full results are in Table C.1.

5 Language-Conditional Molecule Generation

Molecule–language modeling is bidirectional between molecular structures and natural-language descriptions, including both perception and generation. This section showcases large-scale training on our curated dataset for language-conditional molecule generation: given a detailed and unambiguous structural description, the model is to generate the exact corresponding molecular structure.

One direct application of this model is integration with chemical drawing software, allowing users to create structures from language or voice instructions instead of drawing atoms and bonds manually. More importantly, such a model could support chemical design, optimization, and synthesis planning in LLM-centric AI systems by translating design intent into concrete molecular structures.

5.1 Experimental Setup

Representing molecular structures as SMILES allows a language-only LLM to handle this multimodal task. We train a LoRA adapter on Qwen3.5-4B (Qwen Team, 2026a) using reinforcement learning (RL) to guide the model toward efficient, accurate reasoning for molecule generation. Since RL training for a small LLM can be difficult to stabilize when the initial policy is weak, we first perform supervised fine-tuning (SFT) before RL, following common practice for RL training of small models (Guo et al., 2025). We use the same prompt as in the LLM-based reconstruction validation. Our curated dataset contains 163,111 samples. Of these, 2,000 samples were validated (Sec. 4.2) and are reserved for later evaluation, while the remaining 161,111 samples form the training set.

Supervised fine-tuning Using the training set, we collect outputs from GLM-5.1 and Qwen3.6-35B for the same language-conditional molecule generation task. Among the generated outputs, 80,850 exactly match the target molecules. After removing samples with reasoning traces longer than 40,960 tokens, we retain 75,666 samples. We use both their reasoning traces and final answers as the SFT training corpus. The adapter is fine-tuned for 3 epochs, taking approximately 19.1 h on 8 H100 GPUs. Appendix C.1 provides further details on the SFT dataset and training configurations.

Reinforcement learning After SFT, we apply large-scale RL using the decoupled clip and dynamic sampling policy optimization (DAPO) (Yu et al., 2025). The reward function rewards correctly formatted outputs. It also provides tiered rewards based on whether the SMILES is valid, similar to, or exactly matches the ground-truth, and applies a soft penalty to overlong reasoning traces. We conduct RL on the full 161,111-sample training set for 1,500 rollout steps. At each rollout step, we sample 512 prompts and generate 8 responses per prompt for group-based reward, followed by 16 optimization steps. The full RL training takes about 322.6 h on two DGX H100 nodes with 16 GPUs in total. More details on the reward formula and RL settings are provided in Appendix C.2.

5.2 Experimental Results

Of the 2,000 validated samples, 1,972 pass our validation and form the MolLangData test set. We also evaluate on the expert-written and validated MolLangBench core and extended sets (Cai et al.,

2026), each containing 200 in-the-wild descriptions of molecules with no overlap with our training set. We evaluate Qwen3.5-4B at different training stages and compare it with 16 much larger LLMs from the GPT (OpenAI, 2026a), Gemini (Google, 2026), DeepSeek (Xu et al., 2026), Qwen (Qwen Team, 2026a), and GLM (GLM-5-Team, 2026) series [1]. Fig. 3 shows our training trajectory on the MolLangData test set, together with representative LLMs, in terms of exact-match rate to the target molecule and average output tokens, including reasoning tokens. Appendix Table C.1 reports the full results for all evaluated models and evaluation sets, including validity and similarity metrics.

The base Qwen3.5-4B performs poorly on this task. On our test set, only 27.33% of its outputs are valid SMILES and only 12.53% exactly match the targets, while using about 20k output tokens on average. After SFT using outputs from stronger LLMs, exact-match rate increases to 59.84%, while the average output tokens decreases to about 14k. This result already exceeds several larger LLMs and approaches frontier models such as Qwen3.8-Max, GLM-5.2, and DeepSeek-V4-Pro. RL further improves exact-match rate to 84.99%. Despite its much smaller size, the post-RL model outperforms all evaluated models except GPT-5.5 and GPT-5.6 Sol. RL also greatly reduces the length of the reasoning by enforcing the overlong output penalty: the average output length decreases to only $1{,}264$ tokens, corresponding to about a $10\times$ reduction compared with the non-GPT baselines. Its token usage is comparable to GPT-5.6 models, still far more efficient considering the model size.

The trained model also generalizes well to the in-the-wild MolLangBench sets, though exact-match rates are lower than on MolLangData. On the core and extended sets, exact-match rate improves from 12.5%/17.0% before training to 47.0%/57.0% after SFT, and further to 65.0%/72.0% after RL. The RL-trained model shows similar reductions in output tokens on both MolLangBench sets.

Additionally, high exact-match rates of frontier models such as GPT-5.6 Sol and substantial gains from training a small model directly on MolLangData, further support the quality of our descriptions.

6 Additional Use Cases, Challenges, and Future Perspectives

Although frontier LLMs have much improved on structure-grounded tasks, the way they interact with molecular structures differs fundamentally from that of chemists. LLMs rely on extensive knowledge of SMILES syntax and chemical nomenclature, and require substantial reasoning over these rules to process the structure. Even the most token-efficient models for structure generation, our post-RL model and GPT-5.6 Sol, still use around 1k tokens, with most spent enumerating atoms and applying rules to construct SMILES, as shown in Appendix Fig. C.1. Chemists, by contrast, typically recognize structures and draw them directly, without explicitly reasoning through the linear notation. Molecule–language modeling should therefore aim not only for higher accuracy, but also for more efficient and cognitively natural ways to perceive and generate molecular structures.

A more elegant direction is to couple language models with specialized graph-based generators having only tens of millions of parameters (Qin et al., 2025). However, long and fine-grained descriptions (Appendix B.8) make direct generation of complete molecular graphs challenging. LLMs can condense the descriptions without compromising completeness and fidelity, which Appendix B.10 shows is feasible on our dataset, methodologically and economically. Condensation alone is only a mitigation. A potential solution is to introduce intermediate subgraph supervision, instead of training only on the final molecular graph. One could use an LLM to generate such step-by-step subgraph construction traces from our descriptions, with the target molecule used to verify the final graph.

For structure perception, once an automatic verifier, such as the introduced generation model, is reliable, a molecule–language model could similarly be trained with RL. For tasks requiring only structural information, a practical compromise is to generate descriptions on demand using our annotation pipeline, without cross-modal training. Preliminary experiments in Appendix D.1 show that these descriptions help LLMs understand molecular structures and improve downstream property prediction. The descriptions also provide complementary structural information for text-based molecule retrieval, as demonstrated by a simple retrieval system in Appendix D.2.

While MolLangData reflects the chemical space of its sample pool, PubChem, it covers common fragments and connectivity patterns, and our pipeline can scale beyond it. The dataset and cross-modal training provide a foundation for aligning language with molecules, or, more broadly, graphs.

[1] Claude was evaluated, but safety classifiers refused some prompts; partial results are in Appendix C.3.

AI USE STATEMENT

In this work, we used generative AI tools for synthetic data generation, initial data validation, and partial code implementation. We used generative AI to polish the language of the *final* manuscript, limited to linguistic aspects only. We did not use LLMs to organize the manuscript or draft any sentences from scratch. The research ideas and method design were developed entirely by the authors. We did not use generative AI to propose the central scientific contributions of this work. We describe the scope of AI use in MolLangData curation and language-conditional molecule generation below.

For MolLangData curation and validation:

- We used LLMs to generate the molecular structure descriptions in MolLangData, as reported in the paper.
- We used LLMs in the initial stage of dataset validation, as reported in the paper.
- We wrote the core rule-based logic for extending OPSIN and constructing enriched XML metadata case by case, with limited AI assistance, and reviewed this code line by line. We used AI coding agents, specifically ChatGPT Codex, to assist with peripheral code related to compilation, testing, and code organization; the authors tested this code for functional correctness.

For the language-conditional molecule generation experiments:

- We evaluated 16 LLMs as baselines on the generation benchmarks, as reported in the paper.
- We used LLM-generated outputs to construct the supervised fine-tuning data, as reported in the paper.
- The training code was developed with substantial assistance from AI coding agents, specifically ChatGPT Codex, following detailed instructions from the authors. The authors conducted extensive, repeated testing and debugging, provided feedback to the agents, and verified that the code behaved as intended.

We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the alignment of molecular structures and natural language. The chemicals in our curated dataset are sourced from publicly available datasets, and the curated descriptions only faithfully describe the structure of the chemicals. Within this scope, the authors declare that no specific ethical concerns need to be reported.

REPRODUCIBILITY STATEMENT

To support reproducibility of the core contributions of this work, we provide code, datasets, and model checkpoints in anonymous repositories. Some artifacts are also included in the Supplementary Material for convenience; others are available only through the repositories due to file size limits. We list the artifacts and their host sites below for MolLangData and for our language-conditional molecule generation experiments, which we call LangMolDiode.

For MolLangData curation and validation:

- The annotation pipeline is available at `https://anonymous.4open.science/r/MolLangData/`. It includes source code for dataset preprocessing and LLM job submission to generate structure descriptions, along with a compiled executable for enriched XML metadata construction. These artifacts are also included in the Supplementary Material.
- The Java source code for the structural metadata generation tool is available at `https://anonymous.4open.science/r/mollangdata_customized_opsin/` and is also included in the Supplementary Material.
- The complete dataset, including approximately 161k training samples and 2k validation samples along with their validation results, is available at `https://huggingface.co/datasets/mollangdata/MolLangData` and is also included in the Supplementary Material.

For the LangMolDiode experiments:

- The source code for training Qwen3.5-4B, including both SFT and RL, is available at `https://anonymous.4open.science/r/LangMolDiode/` and is also included in the Supplementary Material.
- The SFT training corpus, including reasoning traces and final outputs, is available at `https://huggingface.co/datasets/mollangdata/LangMolDiode-SFT-Corpus`.
- The adapter checkpoints after SFT and RL are available at `https://huggingface.co/collections/mollangdata/langmoldiode-checkpoints`.

Due to the inherent stochasticity of LLM API calls, the generated descriptions and validation results may not be reproduced exactly. Our validation on 2k samples provides an estimate of description precision, and we expect regenerated descriptions to achieve similar precision. Likewise, we expect repeated language-conditional molecule generation evaluations, for both the baseline LLMs and our trained models, to fall within a reasonable range of the results reported in this work.

REFERENCES

Derek T Ahneman, Jesús G Estrada, Shishi Lin, Spencer D Dreher, and Abigail G Doyle. Predicting reaction performance in c–n cross-coupling using machine learning. *Science*, 2018.

Anthropic. Introducing claude fable 5 and claude mythos 5. `https://platform.claude.com/docs/en/models/fable-5/introducing-claude-fable-5-and-claude-mythos-5`, 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*, 2019.

Feiyang Cai, Katelin Zacour, Tianyu Zhu, Tzuen-Rong Tzeng, Yongping Duan, Ling Liu, Srikanth Pilla, Gang Li, and Feng Luo. Chemfm as a scaling law guided foundation model pre-trained on informative chemicals. *Communications Chemistry*, 2025.

Feiyang Cai, Jiahui Bai, Tao Tang, Guijuan He, Joshua Luo, Tianyu Zhu, Srikanth Pilla, Gang Li, Ling Liu, and Feng Luo. Mollangbench: A comprehensive benchmark for language-prompted molecular structure recognition, editing, and generation. In *14th International Conference on Learning Representations*, 2026.

He Cao, Zijing Liu, Xingyu Lu, Yuan Yao, and Yu Li. Instructmol: Multi-modal integration for building a versatile and reliable molecular assistant in drug discovery. In *31st International Conference on Computational Linguistics*, 2025.

Yongqiang Chen, Quanming Yao, Juzheng Zhang, James Cheng, and Yatao Bian. Hierarchical graph tokenization for molecule-language alignment. In *42nd International Conference on Machine Learning*, 2025.

Inc. Daylight Chemical Information Systems. Smarts-a language for describing molecular patterns. 2019.

Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 2007.

Carl Edwards, ChengXiang Zhai, and Heng Ji. Text2mol: Cross-modal molecule retrieval with natural language queries. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

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

Carl Edwards, Qingyun Wang, Lawrence Zhao, and Heng Ji. L+ m-24: Building a dataset for language+ molecules@ acl 2024. In *Proceedings of the 1st Workshop on Language+ Molecules (L+ M 2024)*, 2024.

Michel F Eichelbaum, Bernard Testa, and Andrew Somogyi. *Stereochemical aspects of drug action and disposition*, volume 153. Springer Science & Business Media, 2012.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. In *12th International Conference on Learning Representations*, 2024.

Henri A Favre and Warren H Powell. *Nomenclature of organic chemistry: IUPAC recommendations and preferred names 2013*. Royal Society of Chemistry, 2013.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In *36th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

GLM-5-Team. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Google. Introducing gemini 3.7 flash. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/introducing-gemini-3-7-flash/`, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 2025.

Taicheng Guo, Kehan Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. In *37th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. Inchi-the worldwide chemical structure identifier standard. *Journal of cheminformatics*, 2013.

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 ultralarge-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 2020.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 2022.

Kevin Maik Jablonka, Philippe Schwaller, Andres Ortega-Guerrero, and Berend Smit. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, 2024.

Dongki Kim, Wonbin Lee, and Sung Ju Hwang. Mol-llama: Towards general understanding of molecules in large molecular language model. In *39th Conference on Neural Information Processing Systems*, 2025.

Sunghwan Kim, Paul A Thiessen, Evan E Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A Shoemaker, et al. Pubchem substance and compound databases. *Nucleic acids research*, 2016.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *29th ACM SIGOPS Symposium on Operating Systems Principles*, 2023.

Greg Landrum. Rdkit documentation. *Release*, 2013.

Gang Liu, Michael Sun, Wojciech Matusik, Meng Jiang, and Jie Chen. Multimodal large language models for inverse molecular design with retrosynthetic planning. In *13th International Conference on Learning Representations*, 2025.

Daniel Lowe. Chemical reactions from us patents. https://doi.org/10.6084/m9.figshare.5104873.v1, 2017.

Daniel M Lowe, Peter T Corbett, Peter Murray-Rust, and Robert C Glen. Chemical name to structure: Opsin, an open source solution, 2011.

Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Martiño Ríos-García, Benedict Emoekabu, Aswanth Krishnan, Tanya Gupta, Mara Schilling-Wilhelmi, Macjonathan Okereke, Anagha Aneesh, et al. A framework for evaluating the chemical knowledge and reasoning abilities of large language models against the expertise of chemists. *Nature Chemistry*, 2025.

Peter Murray-Rust and Henry S Rzepa. Chemical markup, xml, and the worldwide web. 1. basic principles. *Journal of Chemical Information and Computer Sciences*, 1999.

OpenAI. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2024.

OpenAI. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/, 2025a.

OpenAI. Introducing gpt-5.2. https://openai.com/index/introducing-gpt-5-2/, 2025b.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026a.

OpenAI. Introducing chatgpt images 2.0. https://openai.com/index/introducing-chatgpt-images-2-0/, 2026b.

OpenEye Scientific Software. LexiChem Toolkit. https://www.eyesopen.com/lexichem-tk.

Long Phan and Anna Sztyber-Betley. A benchmark of expert-level academic questions to assess ai capabilities. *Nature*, 2026.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, 2020.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *42nd International Conference on Machine Learning*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, 2026b.

Brandon J Reizman, Yi-Ming Wang, Stephen L Buchwald, and Klavs F Jensen. Suzuki–miyaura cross-coupling optimization enabled by automated feedback. *Reaction chemistry & engineering*, 2016.

Jean-Louis Reymond. The chemical space project. *Accounts of chemical research*, 2015.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *36th Conference on Neural Information Processing Systems*, 2022.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *20th European Conference on Computer Systems*, 2025.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

M.B. Smith. *March's Advanced Organic Chemistry: Reactions, Mechanisms, and Structure*. Wiley, 2020. ISBN 9781119371809.

Bing Su, Dazhao Du, Zhao Yang, Yujie Zhou, Jiangmeng Li, Anyi Rao, Hao Sun, Zhiwu Lu, and Ji-Rong Wen. A molecular multimodal foundation model associating molecule graphs with natural language. *arXiv preprint arXiv:2209.05481*, 2022.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 1988.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 2018.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Zaifei Yang, Hong Chang, RuiBing Hou, Shiguang Shan, and Xilin Chen. Knowmol: Advancing molecular large language models with multi-level chemical knowledge. In *39th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *39th Conference on Neural Information Processing Systems*, 2025.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, et al. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024.

Situo Zhang, Hanqi Li, Lu Chen, Zihan Zhao, Xuanze Lin, Zichen Zhu, Bo Chen, Xin Chen, and Kai Yu. Reasoning-driven retrosynthesis prediction with large language models via reinforcement learning. *arXiv preprint arXiv:2507.17448*, 2025.

Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhi-Hong Deng, Lingpeng Kong, and Qi Liu. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. In *37th Conference on Neural Information Processing Systems*, 2023.

Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Yi Xia, Bo Chen, Hongshen Xu, Zichen Zhu, Su Zhu, et al. Developing chemdfm as a large language foundation model for chemistry. *Cell Reports Physical Science*, 2025.

Chenyue Zhou, Mingxuan Wang, Yanbiao Ma, Chenxu Wu, Wanyi Chen, Zhe Qian, Xinyu Liu, Yiwei Zhang, Junhao Wang, Hengbo Xu, et al. From perception to cognition: A survey of vision-language interactive reasoning in multimodal large language models. *arXiv preprint arXiv:2509.25373*, 2025.

Yihan Zhu, Gang Liu, Eric Inae, and Meng Jiang. Moltextnet: A two-million molecule-text dataset for multimodal molecular learning. *arXiv preprint arXiv:2506.00009*, 2025.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A Related Works **16**

B Additional Information for Dataset Curation and Validation **17**

B.1 Additional Information for the Illustrative Example in Figure 2 17

B.2 Dataset Prefiltering Statistics 18

B.3 Generation Difficulty Categories 18

B.4 Ablation Study 18

B.5 Model Snapshots 20

B.6 Validation Statistics 20

B.7 Empirical Evaluation of False-Positive Rate in LLM Reconstruction 21

B.8 Dataset Statistics 21

B.9 Token Usage and Cost Estimation 21

B.10 Feasibility of Description Condensation 21

C Supplementary Information for Language-Conditional Molecule Generation **23**

C.1 Supervised Fine-tuning 23

C.2 Reinforcement Learning 23

C.3 Benchmark Results 24

C.4 Example of Molecule Generation Reasoning 24

D Additional Experiments for Dataset Usage **26**

D.1 Downstream Task Evaluation of Description Generation Pipeline 26

D.2 Preliminary Demonstration of Text-Based Molecular Retrieval 27

E Prompts and Representative Examples **28**

E.1 Prompt for Language-Conditional Molecule Generation 28

E.2 Description Generation Prompts and Complex-Ring XML Semantics 28

E.3 Representative Samples 36

A Related Works

Molecule–language cross-modal research predates the LLM era. Pilot works such as Text2Mol (Edwards et al., 2021) and MolT5 (Edwards et al., 2022) curate paired language descriptions and molecules from public chemical databases such as PubChem (Kim et al., 2016) and ChEBI (Degtyarenko et al., 2007), where the descriptions primarily cover physicochemical and pharmacological properties with only coarse structural information. These datasets are then used to train models to translate such descriptions into molecular structures.

With the emergence of LLMs, their remarkable capabilities have motivated applications across diverse scientific disciplines, including chemistry. Researchers have sought to leverage the flexibility, reasoning capacity, and extensive knowledge embedded in LLMs to tackle general chemistry problems, as evaluated by benchmarks such as ChemBench (Mirza et al., 2025) and Humanity's Last Exam (Phan & Sztyber-Betley, 2026). LLMs have also been explored for a range of molecular tasks long studied in the machine learning community, including property prediction (Wu et al., 2018), synthesis prediction and planning (Lowe, 2017), reaction yield prediction (Reizman et al., 2016; Ahneman et al., 2018), molecular optimization (Gao et al., 2022), and *de novo* molecular generation (Brown et al., 2019; Polykovskiy et al., 2020). These tasks and their associated datasets have been rephrased into natural-language form, with efforts including benchmarking LLM performance (Guo et al., 2023) and curating instruction-tuning datasets (Fang et al., 2024). This has led to the development of molecule–language multi-modal models, including general-purpose models targeting multiple chemical tasks and chemical question answering (Cao et al., 2025; Zhang et al., 2024; Zhao et al., 2025; Chen et al., 2025), and task-specific models for property prediction (Zhao et al., 2023), molecular design (Liu et al., 2025), and retrosynthesis prediction (Zhang et al., 2025).

Such models have yet to demonstrate compelling performance on par with traditional uni-modal chemical models (Cai et al., 2025). We argue that the underlying cause is the gap in molecular structure–language alignment. As benchmarked by MolLangBench (Cai et al., 2026), even state-of-the-art LLMs remain far from perfect on intuitive tasks such as molecular structure recognition and manipulation. Since chemical reasoning always begins with accurate structure understanding for chemists, so should AI models. Without reliable structure–language alignment, the powerful reasoning capacity and rich knowledge of LLMs cannot be effectively exploited, nor can downstream performance be expected to improve.

Encouragingly, several recent works have begun to recognize that structure–language alignment is central to molecule–language cross-modal modeling. MolTextNet (Zhu et al., 2025) extracts structural features using RDKit (Landrum, 2013), relying on 90 functional groups defined by SMARTS patterns, but this captures only the presence of an incomplete set of functional groups, losing important structural information such as stereochemistry, ring topology, and substituent positioning. MolLLaMA (Kim et al., 2025) prompts GPT-4o with IUPAC names to generate structure-level molecular descriptions. Similarly, KnowMol (Yang et al., 2025) combines both ideas, first identifying 82 common functional groups using RDKit and then feeding the detected groups together with SMILES and IUPAC names into GPT-4o to generate structural descriptions. Besides the incompleteness of these descriptions, the fundamental issue is reliability: as both we and MolLangBench (Cai et al., 2026) demonstrate, even models considerably stronger than GPT-4o still fail to consistently produce accurate structural descriptions, as confirmed by manual inspection of their released datasets. In contrast, this work uses a rule-based approach to guide reliable description generation, producing structurally unambiguous descriptions that preserve complete structural details, such that the full molecule can be reconstructed from the description alone.

B Additional Information for Dataset Curation and Validation

B.1 Additional Information for the Illustrative Example in Figure 2

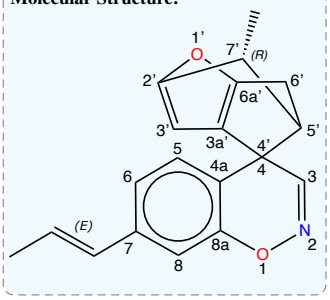
Generated Description:

Build a spiro compound in which one tetrahedral carbon is shared by two different ring systems (a "spiro" junction).

Unprimed ring system (benzo-fused 1,2-oxazine). Define a fused bicyclic framework and label its atoms 1, 2, 3, 4, 4a, 5, 6, 7, 8, and 8a as follows. Make a six-member 1,2-oxazine ring containing adjacent heteroatoms: atom 1 is an oxygen (O1) and atom 2 is a nitrogen (N2). Connect O1–N2 by a single bond, then make N2 double-bonded to carbon atom 3 (C3). Bond C3 by a single bond to carbon atom 4 (C4), and bond C4 by a single bond to carbon atom 4a (C4a). Close the oxazine ring by connecting C4a to carbon atom 8a (C8a) (this C4a–C8a bond is the fusion bond to a benzene ring) and then connect C8a back to O1 by a single bond.

Fuse a benzene ring onto the oxazine by using C4a and C8a as the two shared adjacent ring atoms: the benzene ring is the six-member aromatic ring C4a–C5–C6–C7–C8–C8a (with C4a–C8a being the shared edge). At C7, attach a prop-1-en-1-yl substituent: the ring carbon C7 is bonded to an sp2 carbon (call it P1) that is double-bonded to a second alkene carbon (P2), and P2 is single-bonded to a terminal methyl carbon (P3). Specify the alkene geometry as E, meaning the higher-priority substituents on P1 and P2 (the ring at P1 and the methyl P3 at P2) lie on opposite sides of the P1=P2 double bond.

Molecular Structure:



Primed ring system (a [2,5]-methano cyclopenta[b]furan). Label this ring system with primes as 1', 2', 3', 3a', 4', 5', 6', 6a', and 7'. First make a five-member furan-type ring: O1'–C2'=C3'–C3a'=C6a'–O1', with double bonds specifically between C2' and C3', and between C3a' and C6a'. Next, fuse a five-member carbocycle onto that furan by sharing the C3a'–C6a' edge: complete the carbocycle as C3a'–C4'–C5'–C6'–C6a' (all single bonds along this path, and the C3a'=C6a' bond being the shared fused edge). The "5',6'-dihydro" part is reflected by keeping the C5'–C6' bond as a single (saturated) bond. Add the "methano" bridge carbon C7' as a single carbon that is bonded to both C2' and C5' (so C2'–C7'–C5' is the bridge). Put a methyl group on C7', and state that C7' is a stereocenter of absolute configuration R.

Spiro connection. Make the unprimed atom C4 and the primed atom C4' the same physical carbon atom (the spiro atom). Thus this spiro carbon is bonded to C3 and C4a in the oxazine/benzene system, and to C3a' and C5' in the methanocyclopenta[b]furan system.

Figure B.1: Molecular structure and generated structural description for the illustrative example shown in Fig. 2. The structured metadata provided as input to the LLM are illustrated in Fig. 2.

Native OPSIN-Generated XML Parse Tree for the Illustrative Example:

```
<molecule name="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]oxazi
↪  ne-4,4'-[2,5]methanocyclopenta[b]furan]">
 <wordRule wordRule="simple" type="full"
 ↪  value="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]ox...">
  <word type="full"
  ↪  value="(7'R)-7'-methyl-7-((E)-prop-1-en-1-yl)-5',6'-dihydrospiro[benzo[e][1,2]ox...">
   <substituent locant="7'">
    <group type="chain" subType="alkaneStem" value="C" labels="numeric">meth</group>
    <suffix type="inline" value="yl">yl</suffix>
    <hyphen value="-">-</hyphen>
   </substituent>
   <bracket locant="7">
    <substituent>
     <group type="chain" subType="alkaneStem" value="CCC" labels="numeric">prop</group>
     <suffix type="inline" value="yl" locant="1">yl</suffix>
    </substituent>
    <hyphen value="-">-</hyphen>
   </bracket>
   <root>
    <group type="ring" subType="ring" value="pent[b]furan"
    ↪  labels="1/2/3/4/5">spirobenzo[e]oxazinpent[b]furan</group>
   </root>
  </word>
 </wordRule>
</molecule>
```

Figure B.2: Molecular structure XML parse tree produced by the native OPSIN tool (Lowe et al., 2011) after the structure assembly for the illustrative example shown in Fig.2. The corresponding XML representation generated by our approach is shown in Fig.2 for comparison.

B.2 Dataset Prefiltering Statistics

We randomly sample 200,000 molecules from PubChem (Kim et al., 2016) and apply prefiltering before description generation. The breakdown is as follows:

(i) 28,908 multi-component entries (containing '.' in SMILES) are excluded to keep validation focused on individual molecular structures. OPSIN and our pipeline natively support these entries, and each component can also be processed separately.

(ii) 1,705 entries lack IUPAC names in PubChem.

(iii) 765 entries have mismatches between the PubChem SMILES and the structure reconstructed by OPSIN from the IUPAC name. Most are due to inconsistencies in the PubChem records rather than errors in our pipeline. For example, the PubChem SMILES for CID 136130191 specifies two E/Z configurations, but its IUPAC name specifies only one.

(iv) 1,206 entries produce OPSIN errors or warnings. Some are due to issues in the PubChem IUPAC names, such as ambiguous locants, rather than limitations of OPSIN or our pipeline.

(v) The remaining 167,416 molecules are sent to LLMs for description generation. After atom-matching filtering and excluding incomplete API calls, we obtain 163,111 final descriptions.

Most exclusions are therefore due to our focus on single-component molecules or issues in the PubChem records. Even an overestimate that counts all cases in categories (iii) and (iv) as pipeline failures gives a failure rate of only 0.99% among the 200,000 candidates, which is acceptable given the complexity of IUPAC nomenclature.

The dataset inevitably reflects the chemical space sampled from PubChem rather than the full chemical space. However, molecular structure descriptions focus on fragments, substituents, and connectivity, and our dataset includes diverse examples of these features and connection topologies.

B.3 Generation Difficulty Categories

We classify molecules as easy, medium, or hard based on ring topology and automatically route them to different LLM reasoning effort levels to balance description quality and generation cost.

- **Easy:** No fused ring systems; only isolated rings and/or acyclic chains, including rings sharing a single spiro atom.
- **Medium:** Exactly one fused ring system containing two rings, with no spiro or bridged junctions within that system.
- **Hard:** A fused ring system containing more than two rings, multiple fused ring systems, or fused systems with spiro or bridged connectivity.

These categories indicate generation difficulty but do not necessarily reflect overall molecular complexity. Even molecules in the easy category can be structurally complex, as illustrated in Sec. E.3.

B.4 Ablation Study

We perform ablation studies to analyze the impact of major design choices in our generation pipeline.

Effect of Structure Metadata We ablate structural metadata to evaluate whether injecting this information improves description precision. Specifically, we regenerate descriptions for the same set of molecules used in the atom-matching filtering study, using identical generation models but providing only the SMILES and IUPAC name, without structural metadata (see Appendix Prompt 3).

Removing structural metadata reduces sampling efficiency, as reflected by the fraction of molecules passing the atom-matching check dropping from 97.7% to 93.6%. We then apply the same hybrid validation procedure to samples that pass atom matching (Appendix Table B.1). Although descriptions generated without metadata achieve a relatively high precision of 94.1%, this is approximately 4.5 percentage points below the baseline with metadata. The performance gap increases at higher difficulty levels. Structural metadata explicitly provides ring fusion, bridged and spiro relationships, and ring-system labeling information that otherwise requires interpretation of IUPAC names or SMILES, helping the model generate more accurate and unambiguous descriptions. During validation, we observed that descriptions generated without structural metadata often omit fused-ring information and labeling schemes. Such descriptions are not considered incorrect in validation,

since expert validators can still unambiguously reconstruct the exact structure using their knowledge of chemical nomenclature. However, they are less suitable for structure–language alignment. In contrast, validators confirm that descriptions generated with metadata consistently include these details. The benefit of structural metadata goes beyond the improvement in precision shown in the table: it also makes structural details more explicit in the descriptions.

Effect of Atom-Matching Filtering To evaluate this effect, we validate $2{,}000$ samples with the same difficulty-level distribution as the main validation set. This set largely overlaps with the main validation set reported in Table 1: the main validation set is constructed by taking the passing samples from this ablation set and supplementing with additional passing samples to reach $2{,}000$. Among these samples, 97.7% pass the atom-matching check, indicating that the filter preserves the vast majority of generated data and does not significantly reduce generation efficiency.

We then apply the same hybrid validation procedure described above to both the passing and failing subsets. For samples that pass the atom-matching filter, description precision remains comparable to the main validation results, achieving an overall precision of 98.6%. In contrast, precision drops sharply to 27.7% for the small subset that fails the filter, demonstrating that atom-matching effectively removes incorrectly labeled samples. Detailed results are reported in Table B.1.

We should note that the non-hydrogen atom count is requested together with description generation in the same LLM call (Prompt 2). Since the prompt also provides SMILES as a reference for description generation, this introduces a potential risk of information leakage into the atom count. We explicitly instruct the model to count atoms *only from the generated description and not from the SMILES*. This lightweight check helps detect missing atoms in long side chains and linkers, as supported by the results above. Requesting the count separately, using only the generated description, would make the check independent of the reference SMILES but add substantial cost at our dataset scale. Our shared-prompt design therefore balances filtering effectiveness and generation cost, while description quality is assessed separately through our strict validation protocol.

Table B.1: Validation precision under structural metadata and atom-matching ablations across generation difficulty levels. Each entry is reported as passed / total evaluated samples (precision $\%$).

	With structural metadata		Without structural metadata (atom match)
	Atom match	Atom mismatch	
Easy	$1{,}275/1{,}292$ (98.7%)	$8/25$ (32.0%)	$1{,}191/1{,}242$ (95.9%)
Medium	$472/476$ (99.2%)	$5/20$ (25.0%)	$427/457$ (93.4%)
Hard	$178/185$ (96.2%)	$0/2$ (0.00%)	$142/172$ (82.6%)
Overall	$1{,}925/1{,}953$ (98.6%)	$13/47$ (27.7%)	$1{,}761/1{,}871$ (94.1%)

Sensitivity to Generation Model Description generation requires LLMs to have strong chemical knowledge and the ability to reason over long, information-dense contexts. To study model sensitivity, we replace GPT-5.2 with GPT-5 (OpenAI, 2025a) for description generation on the same molecule set used in prior ablations. To reduce human validation cost, we use GPT-5.2 as the sole validator for both models and evaluate descriptions that pass the atom-matching check. The validation pass rate for GPT-5-generated descriptions is comparable to that of GPT-5.2 in the easy category (0.6 percentage points lower), but is 18.0 and 20.7 percentage points lower in the medium and hard categories, respectively. Detailed results are reported in Table B.2. Because we use only LLM validation, which can produce false negatives, the actual precision gap may be smaller. These results suggest that description reliability is affected by generation model capacity, particularly for complex structures. Other commercial LLMs may perform differently depending on their chemical reasoning capabilities. Our choice of LLMs balances generation quality and budget constraints at the time of dataset curation.

Table B.2: Ablation study on model sensitivity. Pass rates ($\%$) of the LLM-based validator for descriptions generated by GPT-5 and GPT-5.2 across different difficulty levels.

Generation Model	Easy	Medium	Hard
GPT-5	88.9%	66.0%	42.5%
GPT-5.2	89.5%	84.0%	63.2%

B.5 Model Snapshots

We use multiple LLMs for dataset collection and validation, as well as baseline comparisons in the language-conditional molecule generation experiments. We access GPT models through Azure AI Foundry APIs, and `claude-fable-5` (Anthropic, 2026), `gemini-3.7-flash` (Google, 2026), and `qwen3.8-max` (Qwen Team, 2026b) through their official platforms. The remaining models are open-weight models hosted by our institution. Table B.3 lists the model snapshots or checkpoint commit hashes, as applicable.

Table B.3: Model snapshots for data generation, validation, and the conditional generation benchmarks in Table C.1. When the snapshot date is unavailable, we report the release date. For open-weight models, we report the commit hash and date.

Model	Snapshot
Qwen models	
`qwen3.5-4b`	2026-03-01 (commit_hash: 851bf6e806efd8d0a36b00ddf55e13ccb7b8cd0a)
`qwen3.6-35b-a3b-fp8`	2026-04-23 (commit_hash: 95a723d08a9490559dae23d0cff1d9466213d989)
`qwen3.8-27b-fp8`	2026-08-14 (commit_hash: 017b9c7af6b5689d5dd426a76e0bc077eb5ca20a)
`qwen3.8-max`	2026-08-02 (Release date)
GLM models	
`glm-5.1-fp8`	2026-04-07 (commit_hash: 28d85cc22ceeee52340e6ec3399bda31852b117c)
`glm-5.2-fp8`	2026-06-17 (commit_hash: a0b55e88465d1a06afece97bc8d6b366aff39089)
DeepSeek models	
`deepseek-v4-pro`	2026-08-13
Google models	
`gemma-4-31b`	2026-04-02 (commit_hash: 5dc307ed18be972e945cea1bfd3facf5bfd5fa39)
`gemini-3.7-flash`	2026-08-13 (Release date)
Claude models	
`claude-fable-5`	2026-06-09 (Release date)
GPT models	
`gpt-5`	2025-08-07
`gpt-5.2`	2025-12-11
`gpt-5.4`	2026-03-05
`gpt-5.5`	2026-04-23
`gpt-5.6-luna`	2026-07-09 (Release date)
`gpt-5.6-terra`	2026-07-09 (Release date)
`gpt-5.6-sol`	2026-07-09 (Release date)

B.6 Validation Statistics

We report validation statistics for a subset of $2{,}000$ samples across easy, medium, and hard difficulty levels. These samples are separate from the molecules used to develop and refine the XML pipeline. Each LLM reconstruction attempt uses a separate call without conversation history. Human validators work independently, without sharing reconstructions or feedback. Both LLM and human reconstruction use the same exact-match criterion: RDKit parses and sanitizes the reconstructed and ground-truth SMILES, converts them to canonical isomeric SMILES, and compares the resulting strings. This check preserves encoded stereochemistry, formal charges, and isotope information.

Among 1,317/496/187 samples in the easy/medium/hard categories, GPT-5.2 validates 1,181/415/117 samples, GLM-5.1 validates $1{,}050/312/65$ samples, and DeepSeek-V4-Pro validates 1,177/415/113 samples. Of these, $950/262/44$ samples pass all three LLM validators and are accepted as correct, leaving 744 samples for human validation.

Human validators assess whether each description unambiguously specifies a unique molecular structure. Using only the description, they reconstruct the molecule in chemical drawing software and submit its SMILES to our internal validation application for the exact-match check. A sample is approved only if the expert confirms that the description is unambiguous and the reconstructed molecule exactly matches the ground truth.

The first human validator approves 343/230/136 samples, and the remaining cases are forwarded to a second validator. The second validator approves only 7 additional easy samples and no additional medium or hard samples. The final counts are therefore 1,300/492/180, totaling 1,972 correctly described samples.

The validation time for a single human annotator is $13.5\,\mathrm{min}/8.3\,\mathrm{min}/12.2\,\mathrm{min}$ for the easy, medium, and hard categories, respectively, with an overall average of $11.7\,\mathrm{min}$ across all validated samples. This is slightly longer than the approximately $10\,\mathrm{min}$ reported in MolLangBench (Cai et al., 2026). We note that these times are collected automatically by our internal validation application and are intended as a coarse reference rather than a precise measure of annotation efficiency.

B.7 Empirical Evaluation of False-Positive Rate in LLM Reconstruction

To reduce human validation effort, we combine LLM-based and human validation, as described in Secs. 4.2 and B.6. This protocol assumes that LLMs are unlikely to reconstruct the exact target molecule from an incorrect description by chance, so we expect a near-zero false-positive rate. Exact reconstruction requires correctly recovering every structural detail from descriptions averaging more than 250 words for molecules with approximately 27 heavy atoms. These details include chemical groups, connectivity, ring topology, labeling semantics, and stereochemistry, as illustrated by the representative examples in Sec. E.3. Even a small error can change the resulting structure, making an exact match from an incorrect description almost impossible.

Even for the 626 samples accepted by one or two LLMs, but not *all* three, and thus rejected by automatic validation, human validation confirms unambiguous descriptions and exact reconstruction of the target molecules. Of these, 451 are accepted by exactly two models: 304 are rejected only by GLM-5.1, 87 only by GPT-5.2, and 60 only by DeepSeek-V4-Pro. The remaining 175 are accepted by only one model: 93 by GPT-5.2, 24 by GLM-5.1, and 58 by DeepSeek-V4-Pro.

We require successful reconstruction by *all* three LLMs to provide an additional safeguard. To evaluate this criterion, we randomly select 200 samples that pass all three LLM validators for additional human validation. Following the standard protocol, human validators receive only the textual descriptions and reconstruct the molecules from them. All 200 descriptions are confirmed to be unambiguous, and the corresponding molecules are reconstructed exactly, with no false positives observed. These results provide empirical support for the reliability of LLM-based validation step.

B.8 Dataset Statistics

We summarize the distribution statistics of the entire generated dataset across easy, medium, and hard difficulty levels in Fig. B.3. For the easy category, molecules span from 2 to 335 non-hydrogen atoms (mean 24.2), with description lengths ranging from 53 to 1308 words (mean 246.7). The medium category covers molecules with 8 to 274 atoms (mean 27.3) and descriptions of 105 to 981 words (mean 273.0). The hard category exhibits the highest structural and linguistic complexity, with atom counts ranging from 11 to 414 (mean 37.2) and description lengths from 126 to 1358 words (mean 334.9).

B.9 Token Usage and Cost Estimation

Generating a large-scale dataset of molecular structure descriptions incurs substantial LLM token usage and associated API costs. We report token and cost statistics estimated from a representative subset of 2,000 sample generations. For this subset, the average token usage per generation is 3,845.35 input tokens and 17,089.12 output tokens (including 16,555.43 reasoning tokens).

As per the OpenAI API Pricing as of 2026-04-28 ($0.875 per 1M input tokens and $7.00 per 1M output tokens under batch mode), the estimated cost per generation is $0.123. Scaling to the full curated dataset of 163k samples, the total estimated cost is more than $20k.

B.10 Feasibility of Description Condensation

Our dataset contains structurally complex molecules, and we impose no verbosity constraints during description generation. The resulting descriptions are therefore often lengthy, as reported in Ap-

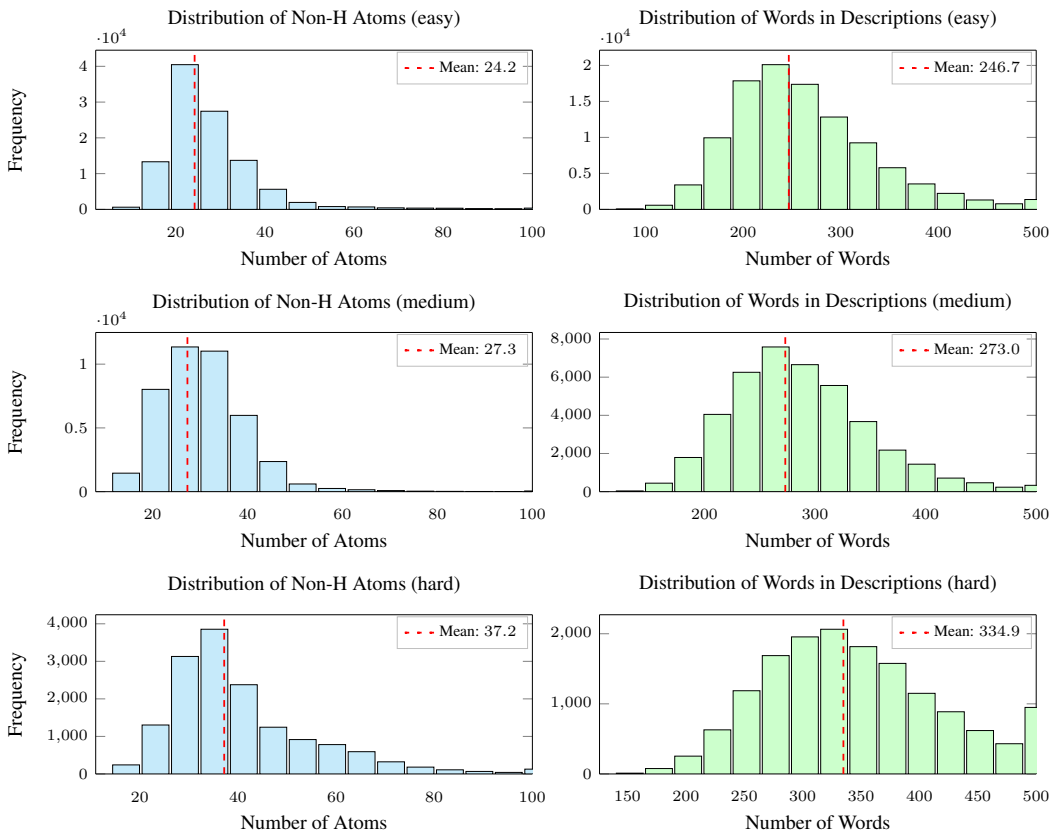


Figure B.3: Distribution statistics of the entire generated dataset, including non-hydrogen atom counts and description word counts, across easy, medium, and hard categories.

pendix B.8. To demonstrate the feasibility of automated condensation, we use GPT-5 with medium reasoning effort to condense descriptions for the same 200 samples used in the false-positive evaluation in Appendix B.7. This reduces the average description length from 249 to 129 words, a 48% reduction. Human validators successfully reconstruct all 200 molecules solely from the condensed descriptions, showing that condensation preserves reconstruction fidelity in these samples. Condensation uses an average of 687 input tokens and 1,757 reasoning and output tokens per sample. At this usage level, condensing the full 163k-sample dataset would cost approximately $1,475.

C Supplementary Information for Language-Conditional Molecule Generation

C.1 Supervised Fine-tuning

As discussed in Sec. 5, we first perform supervised fine-tuning (SFT) because the base Qwen3.5-4B model performs poorly on this task. We use outputs from more capable LLMs, including their reasoning traces, to improve the initial policy and support faster convergence during subsequent RL training. We use the $161{,}111$ MolLangData training samples, which are separate from the $1{,}972$ test samples, to collect molecule-generation outputs from GLM-5.1 and Qwen3.6-35B. Samples are routed to these models according to the availability of our institution's hosted LLM services; not every sample is sent to both models. After deduplication, we obtain $80{,}850$ outputs that exactly match the target molecules: $62{,}286$ from GLM-5.1 and $18{,}564$ from Qwen3.6-35B. We discard outputs longer than $40{,}960$ tokens, leaving $75{,}666$ samples for SFT, comprising $62{,}185$ from GLM-5.1 and $13{,}481$ from Qwen3.6-35B. Both the reasoning traces and final answers are used for training.

We fine-tune Qwen3.5-4B using an 85M-parameter LoRA adapter for 3 epochs, with a batch size of 16 samples per optimization step. Training uses AdamW with a learning rate of 2×10^{-4} and a weight decay of 0.01, following a cosine learning-rate schedule with 20 warm-up steps. SFT takes approximately $19.1\,\mathrm{h}$ on a single DGX H100 node with 8 H100 GPUs, each with $80\,\mathrm{GB}$ of memory. SFT improves the exact-match accuracy of molecule generation on the MolLangData test set from 12.53% to 59.84%, providing a stronger starting point for subsequent RL training.

C.2 Reinforcement Learning

After SFT, we use DAPO (Yu et al., 2025) on the same $161{,}111$ training samples to further improve molecule-generation accuracy and reasoning efficiency.

Reward Design Let $\mathcal{M}$ be the target molecule and $\widehat{\mathcal{M}}$ the molecule parsed from a generated response. We penalize responses with missing or invalid SMILES. For valid predictions, we assign rewards based on structural similarity and exact matching to the target. The base reward R_{base} is

$$R_{\text{base}} = \begin{cases} -1.0, & \text{if no SMILES can be extracted,} \\ -0.4, & \text{if the extracted SMILES is invalid,} \\ \mathrm{sim}(\mathcal{M}, \widehat{\mathcal{M}}) + 0.4 \times \mathbb{1}(\mathcal{M} = \widehat{\mathcal{M}}) + 0.1, & \text{otherwise.} \end{cases}$$

Here, sim is the Tanimoto similarity between Morgan fingerprints of the generated and target. We compute fingerprints with 2048 bits and radius 2. $\mathbb{1}(\mathcal{M} = \widehat{\mathcal{M}})$ evaluates whether their canonical isomeric SMILES match, using the same exact-match criterion as in our dataset validation.

To discourage long reasoning, we apply a linear penalty beyond $L_{\text{min}} = 2{,}048$ response tokens, reaching -0.4 at the maximum length of $L_{\text{max}} = 32{,}768$ tokens:

$$R_{\text{len}}(L) = -0.4 \times \mathrm{clip}\left(\frac{L - L_{\text{min}}}{L_{\text{max}} - L_{\text{min}}}, 0, 1 \right),$$

where L is the response length in tokens. The final reward is

$$R_{\text{final}} = R_{\text{base}} + R_{\text{len}}(L).$$

Training Configuration We implement DAPO using verl (Sheng et al., 2025), with vLLM engines (Kwon et al., 2023) for rollout and the Megatron backend (Shoeybi et al., 2019) for distributed LoRA policy optimization. Each rollout step targets 512 prompt groups, with eight responses per prompt. We compute advantages within each group and use DAPO's token-level clipped policy objective, without a KL-loss term. In practice, all eight responses in a group may receive identical rewards, so their advantages are zero and they provide no policy-gradient signal. Following DAPO's dynamic sampling strategy, we sample 640 prompts, discard groups with identical rewards, and sample additional batches as needed to collect 512 informative groups. The retained groups are divided into 16 mini-batches, each containing 32 prompts and their 256 responses. Each mini-batch is used for one optimization step, giving 16 optimization steps per rollout.

With this sampling procedure, we continue training the LoRA adapter from SFT for $1{,}500$ rollout steps, totaling $24{,}000$ optimizer updates. The learning rate increases linearly from zero to 10^{-6} over the first 20 rollout steps and remains constant thereafter. Training takes approximately $322.6\,\mathrm{h}$ on two DGX H100 nodes with 16 NVIDIA H100 GPUs in total.

C.3 Benchmark Results

We benchmark language-conditional molecule generation on the MolLangData test set ($1{,}972$ samples held out from training) and the MolLangBench core and extended sets (200 samples each) (Cai et al., 2026). We check all three test sets against the training set using the same exact-match criterion based on canonical isomeric SMILES and confirm there is no overlap.

We evaluate Qwen3.5-4B before training, after SFT, and after RL. We compare these results with 16 much larger LLMs, including Qwen, GLM, DeepSeek, Gemini, Claude, and GPT models. Their snapshots or release dates are listed in Table B.3. We extract SMILES from model responses and report validity, along with Tanimoto similarity (invalid outputs scored as 0) and exact-match rate relative to the ground truth. We also report the average number of output tokens to compare token efficiency. Full results are provided in Table C.1. For reference, we also report the best exact-match rate achieved during RL, independently for each test set.

All baseline models use medium reasoning effort except GLM-5.1 and Qwen3.6-35B, which enable reasoning without selectable effort levels, and GLM-5.2, which uses high. Because of frequent API refusals, Claude Fable 5 results use only non-refusal responses and are for reference: $1{,}750$ of $1{,}972$ samples on MolLangData, 196 of 200 on MolLangBench core and all 200 on the extended set.

Table C.1: Language-conditional molecule generation on MolLangData and MolLangBench. Validity and exact-match rate (Exact) are percentages; average output tokens are rounded to integers.

Model	MolLangData (Test; 1,972 Samp.)				MolLangBench (Core; 200 Samp.)				MolLangBench (Extended; 200 Samp.)			
	Validity	Similarity	Exact	Tokens	Validity	Similarity	Exact	Tokens	Validity	Similarity	Exact	Tokens
Qwen models												
`qwen3.6-35b`	71.04%	0.6317	46.42%	51,451	75.00%	0.6379	43.50%	41,832	75.50%	0.6495	47.00%	46,793
`qwen3.8-27b`	76.52%	0.6748	42.75%	8,915	76.50%	0.6717	41.00%	8,956	85.50%	0.7556	53.50%	5,918
`qwen3.8-max`	86.97%	0.8167	63.54%	6,707	82.00%	0.7500	53.00%	6,776	91.50%	0.8478	69.00%	4,578
GLM models												
`glm-5.1-fp8`	70.91%	0.6391	45.57%	13,899	73.00%	0.6401	43.50%	17,091	80.00%	0.7090	53.50%	11,485
`glm-5.2-fp8`	88.87%	0.8351	65.18%	10,912	86.50%	0.7798	53.00%	12,652	91.00%	0.8373	68.00%	7,936
DeepSeek models												
`deepseek-v4-pro`	87.69%	0.8451	68.21%	18,469	89.50%	0.8589	66.00%	22,907	93.50%	0.8988	76.00%	13,335
Google models												
`gemma-4-31b`	58.11%	0.4535	25.91%	12,251	54.00%	0.3801	17.00%	11,159	69.00%	0.5409	30.50%	10,095
`gemini-3.7-flash`	97.82%	0.9571	81.44%	2,527	98.00%	0.9419	75.00%	2,926	99.00%	0.9667	82.50%	2,035
Claude models (partial results; reference only)												
`claude-fable-5`	99.20%	0.9831	92.74%	627	98.98%	0.9823	92.35%	898	99.50%	0.9606	86.50%	550
GPT models												
`gpt-5`	72.46%	0.6490	49.39%	7,539	66.00%	0.5585	38.50%	8,044	76.00%	0.6886	56.00%	6,164
`gpt-5.2`	87.73%	0.8314	68.56%	3,700	89.50%	0.8299	64.00%	4,554	93.00%	0.8649	71.00%	2,514
`gpt-5.4`	94.42%	0.9184	81.95%	3,495	94.50%	0.9016	74.00%	4,274	95.50%	0.9337	84.00%	2,528
`gpt-5.5`	98.38%	0.9756	92.95%	2,623	99.00%	0.9783	93.50%	2,949	100.00%	0.9779	93.50%	1,827
`gpt-5.6-luna`	93.51%	0.9061	78.09%	1,625	92.50%	0.8857	69.50%	1,868	95.00%	0.9114	80.00%	1,242
`gpt-5.6-terra`	92.65%	0.8810	73.48%	1,219	95.00%	0.8777	67.50%	1,328	94.50%	0.8980	76.50%	843
`gpt-5.6-sol`	99.65%	0.9929	94.27%	1,058	99.00%	0.9775	88.50%	1,172	99.50%	0.9824	95.00%	730
Qwen3.5-4B training stages												
pre-SFT; base	27.33%	0.2136	12.53%	19,338	23.50%	0.1985	12.50%	22,241	28.00%	0.2490	17.00%	20,058
post-SFT	84.33%	0.7874	59.84%	14,304	78.00%	0.6924	47.00%	16,362	87.50%	0.7847	57.00%	13,162
post-RL	97.41%	0.9581	84.99%	1,264	93.50%	0.8700	65.00%	1,430	97.50%	0.9155	72.00%	1,067
best-checkpoint	96.91%	0.9551	85.19%	1,277	93.50%	0.8681	71.50%	1,273	97.00%	0.9248	77.50%	1,294

C.4 Example of Molecule Generation Reasoning

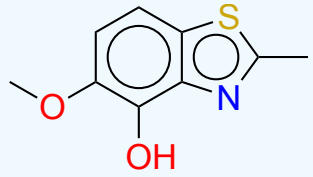
Fig. C.1 shows an example output from our post-RL Qwen3.5-4B model. Although the molecule contains only a two-ring fused system with three substituents, the model uses 590 output tokens to generate its SMILES representation. Much of this output is used to enumerate atom positions and apply SMILES syntax rules to construct and repeatedly check the resulting string.

Generated Description:

Build a bicyclic aromatic **benzothiazole** core (total **nine ring atoms: seven carbons, one nitrogen, and one sulfur**) by fusing a benzene ring to a 1,3-thiazole ring.

For an unambiguous layout, label the fused system as follows: let **position 1** be the **sulfur** in the five-membered thiazole ring. Moving around that thiazole ring, the next atom (**position 2**) is a **carbon**, followed by **position 3** as the **nitrogen**. The next atom is the **ring-junction carbon** shared with the benzene ring (often written as **3a**); from that junction, continue around the benzene portion through four aromatic carbons (**positions 4, 5, 6, and 7**) and return to the second shared junction carbon next to sulfur (often written as **7a**), which closes both rings back to the sulfur at position 1.

Molecular Structure:



Add substituents to this fused aromatic scaffold:

- On the **position-2 carbon** (between S and N), attach a **methyl group** (–CH_3).
- On **position 4** of the benzene portion (the aromatic carbon next to the N-side junction), attach a **hydroxyl group** (–OH).
- On the adjacent benzene carbon, **position 5**, attach a **methoxy group**, i.e., an **–O–CH_3** substituent with the oxygen directly bonded to the ring carbon.

No stereocenters or E/Z geometry are involved.

LLM Output (590 tokens):

The user wants a SMILES representation of a molecule based on a detailed description.

1. **Core structure**: Benzothiazole.
   - Position 1: Sulfur (S)
   - Position 2: Carbon (C) with a methyl group (-CH3)
   - Position 3: Nitrogen (N)
   - Position 3a: Ring-junction carbon (shared with benzene)
   - Position 4: Carbon with hydroxyl group (-OH)
   - Position 5: Carbon with methoxy group (-OCH3)
   - Position 6: Carbon
   - Position 7: Carbon
   - Position 7a: Ring-junction carbon (shared with benzene, next to S)
2. **Layout**:
   - S1 - C2(-CH3) - N3 - C3a - C4(-OH) - C5(-OCH3) - C6 - C7 - C7a - S1
3. **SMILES construction**:
   - Start at S1: s1
   - Go to C2: c(C)
   - Go to N3: n
   - Go to C3a: c2
   - Go to C4: c(O)
   - Go to C5: c(OC)
   - Go to C6: c
   - Go to C7: c
   - Go to C7a: c2
   - Close to S1: 1
   - Let's trace the SMILES:
     - S1: s1
     - C2: c(C)
     - N3: n
     - C3a: c2
     - C4: c(O)
     - C5: c(OC)
     - C6: c
     - C7: c
     - C7a: c2
     - Close S1: 1
   - So: s1c(C)nc2c(O)c(OC)ccc21
4. **Verify the positions**:
   - S1: s1
   - C2: c(C)
   - N3: n
   - C3a: c2
   - C4: c(O)
   - C5: c(OC)
   - C6: c
   - C7: c
   - C7a: c2
   - Close S1: 1
   - Matches perfectly.
5. **Final SMILES**: s1c(C)nc2c(O)c(OC)ccc21

<smiles>COc1ccc2sc(C)nc2c1O</smiles>

Figure C.1: An example output from our post-RL Qwen3.5-4B model.

D Additional Experiments for Dataset Usage

Beyond molecule–language cross-modal training, we briefly explore a few additional, preliminary use cases enabled by our annotation pipeline and curated dataset.

D.1 Downstream Task Evaluation of Description Generation Pipeline

Our annotation pipeline generates accurate and complete molecular structure descriptions, which can be directly used for tasks requiring structural information as input. We demonstrate that the automatically generated structural descriptions can benefit (i) molecular structure recognition, evaluated on the MolLangBench (Cai et al., 2026) benchmark, and (ii) downstream chemical property reasoning, evaluated on the MoleculeNet (Wu et al., 2018) benchmark. For both benchmarks, molecular structures are first converted to IUPAC names using the OpenEye LexiChem Toolkit (OpenEye Scientific Software), which are then passed to our pipeline for structural description generation.

Molecular structure recognition. We select 10 molecular recognition subtasks from MolLangBench that do not require identifying specific target atoms (such as identifying k-hop neighbors, determining bond connectivity, or characterizing the specified atom or bond stereochemistry), as such tasks would require manually specifying atoms in natural language when using descriptions as input. Localization accuracy is also excluded for the same reason. For each query, we generate a structural description using our pipeline and prompt GPT-5, the best-performing model evaluated in MolLangBench, at medium reasoning effort to perform the recognition task solely based on the description.

The results are reported in Table D.1. Despite SMILES representations already achieving strong performance at 0.963 average accuracy, our generated structural descriptions match or improve performance on 9 out of 10 subtasks (3 retain perfect scores of 1.000 and 6 improve), reaching 0.985 average accuracy. Notably, ring junction recognition (i.e., atoms shared between two or more rings) improves by 0.105, suggesting that structural descriptions effectively assist LLMs in understanding molecular topology. These results demonstrate that descriptions generated by our pipeline can enhance structural understanding in LLM-centric AI systems.

Table D.1: Molecular structure recognition results on MolLangBench (Cai et al., 2026), comparing SMILES representations against structural descriptions generated by our pipeline. Both are evaluated using GPT-5 with medium reasoning effort; the SMILES baseline results are taken directly from MolLangBench.

Task	SMILES	Structural Description	Δ
Aldehyde	1.000	1.000	0.000
Benzene	0.925	0.995	+0.070
Carboxyl	0.980	0.985	+0.005
Furan	0.995	1.000	+0.005
Halogen atoms	1.000	1.000	0.000
Ketone	1.000	1.000	0.000
Pyridine	0.915	0.955	+0.040
Quaternary carbons	0.980	1.000	+0.020
Ring junctions	0.830	0.935	+0.105
Thiophene	1.000	0.985	−0.015
Average	0.963	0.985	+0.022

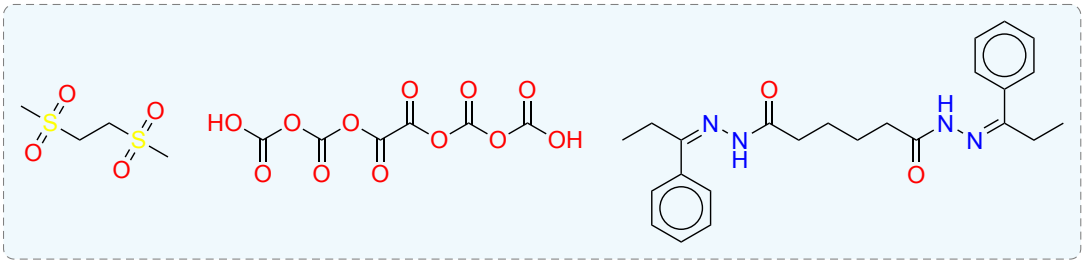
Molecular property prediction. To demonstrate that improved structural understanding further enhances downstream chemical reasoning, we evaluate on two tasks from the MoleculeNet benchmark: BACE, a binary classification task for inhibitors of human β-secretase 1, and ESOL, a regression task for water solubility with continuous labels in $\log \mathrm{mol/L}$. We compare two molecular representations as input: SMILES and our generated structural descriptions, both evaluated with GPT-5 using medium reasoning effort. Note that GPT-5 is a reasoning model that internally analyzes molecular structure even when given only SMILES as input.

Following MoleculeNet, we use scaffold splits for BACE and random splits for ESOL. Results are reported in Table D.2 as mean ± standard deviation across three splits. Our structural descriptions improve performance by 5.03% on BACE and 11.97% on ESOL, directly demonstrating that our pipeline enhances LLM-based chemical reasoning. We expect the advantage to be even larger in tasks requiring fine-grained structural information, such as forward/retro-synthesis prediction or molecular property optimization.

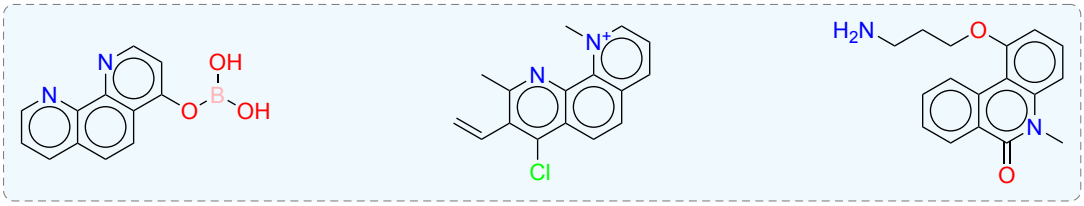
Table D.2: Molecular property prediction results on the BACE and ESOL datasets from MoleculeNet (Wu et al., 2018), comparing SMILES representations against structural descriptions generated by our pipeline. Both are evaluated using GPT-5 with medium reasoning effort. Values are mean ± standard deviation across three splits, with bold indicating the better performance.

Dataset	SMILES	Structural Description
BACE (Accuracy ↑)	0.656 ± 0.014	$\mathbf{0.689 \pm 0.016}$
ESOL (RMSE ↓)	0.844 ± 0.024	$\mathbf{0.743 \pm 0.073}$

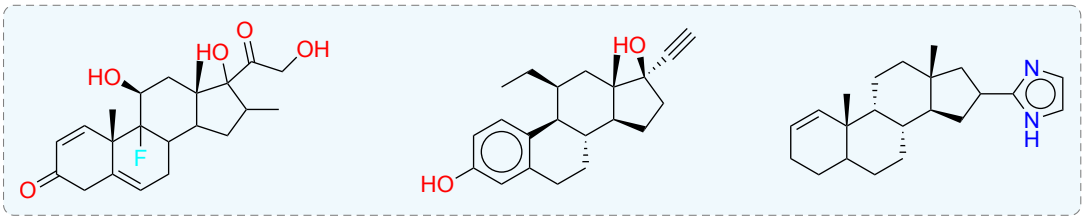
D.2 Preliminary Demonstration of Text-Based Molecular Retrieval



Text query 1: *"a molecule with a symmetric structure"*



Text query 2: *"an angular fused tricyclic aromatic system"*



Text query 3: *"a molecule with a steroid-like fused ring system"*

Figure D.1: Text-based molecular retrieval over our 163k generated descriptions. Each row shows the top-3 retrieved molecules for the corresponding natural-language structural query shown below.

Descriptions in current public chemical repositories, such as PubChem (Kim et al., 2016) and ChEBI (Degtyarenko et al., 2007), contain only coarse structural information and physicochemical or pharmacological properties. Our generated descriptions, which capture complete structural details, can serve as complementary information for textual molecule retrieval systems.

We provide a preliminary demonstration using a simple dense retrieval system over our 163k generated descriptions. The index combines full-description embeddings with chunk-level embeddings. At the full-description level, we precompute one embedding per molecule using text-embedding-3-large (OpenAI, 2024). At the chunk level, each description is split into overlapping 50-word chunks with 20-word overlap, and each chunk is independently embedded using the same model. At query time, the user query is embedded and searched against both indices. Chunk-level matches are aggregated back to the molecule level by taking the best chunk score for each molecule, with a small bonus when multiple chunks from the same molecule are retrieved. The final ranking is obtained by combining the full-description and chunk-level rankings using weighted reciprocal rank fusion.

Fig. D.1 shows the top-3 retrieval results for three example queries of different structural specificity. The retrieved molecules are consistent with the intended structural characteristics described in each query, suggesting that our generated descriptions can support meaningful text-based molecular retrieval. Notably, the first two queries describe structural patterns that are difficult to express as SMARTS (Daylight Chemical Information Systems, 2019) regular expressions. We emphasize that this is a preliminary demonstration; more sophisticated retrieval architectures and systematic evaluation are left for future work.

E Prompts and Representative Examples

E.1 Prompt for Language-Conditional Molecule Generation

We use the prompt from Appendix A.21.3 of MolLangBench (Cai et al., 2026), listed below, to ask an LLM to generate a molecule in SMILES format from a detailed language description. This prompt is used both for LLM-based validation in Sec. 4.2 and for the language-conditional molecule generation experiments in Sec. 5.

Prompt 1: Molecule Generation from Natural-Language Descriptions

Instructions:

Analyze the given molecule structure description and generate the corresponding molecular structure as SMILES representation.

- This description provides an unambiguous and detailed account sufficient to completely reconstruct the molecule.
- Interpret stereochemistry, charges, and isotopes if mentioned in the description.
- Return the resulting SMILES string enclosed within <smiles> and </smiles> tags.

Molecule structure description:

`{Structure Description}`

E.2 Description Generation Prompts and Complex-Ring XML Semantics

Prompt 2: Molecular Description Generation from SMILES, IUPAC Names, and Structural Metadata

Task:

You are provided with the **IUPAC name**, **SMILES string**, and **rule-based hierarchical metadata** describing the molecular structure. Generate a **detailed and accurate structural description** that would allow a person with **basic organic chemistry knowledge** to reconstruct the exact molecule using only your structural description text. The description should be **natural**, **diverse**, and **chemically precise**, clearly conveying the molecular skeleton, substituents, and stereochemistry.

In addition, after completing the description, report the total number of non-hydrogen atoms implied by your description, computed **only from the information explicitly stated in the description itself**.

Guidelines:

1. **Purpose and Independence**

- The description must be **self-contained** and **sufficient for reconstruction**.
- Assume the reader only has your final text—they will **not see** the SMILES, IUPAC name, or metadata.
- You may use all input data internally to reason about the molecule, but the final description must read as a stand-alone, human-readable explanation.

2. **Freedom of Description**
   You may begin from any perspective—the main skeleton, a key ring system, or an important substituent. Combine information freely from the IUPAC name, SMILES, and metadata to capture the complete structure.
3. **Backbone and Connectivity**
   Describe how rings, chains, and substituents are connected. Indicate branching positions, linkages, and overall topology so that the structure can be reconstructed accurately.
4. **Functional Groups and Substituents**
   Identify all key functional groups and substituents. Specify their type (e.g., hydroxyl, amine, halogen, carbonyl), location, and bonding pattern relative to the molecular framework.
5. **Simple or Isolated Rings**
   For common rings (benzene, pyridine, cyclohexane), you may name them directly or briefly describe their composition and bonding.
6. **Fused, Bridged, or Spiro Ring Systems — explicit, structured, and verified**
   It is **strongly recommended to follow the guidance below** when describing complex ring topologies:
   When interpreting fused, bridged, or spiro ring systems, you should explicitly refer to the corresponding ring-system semantics described later. These semantics are essential for correctly understanding atom labeling, fusion points, bridge connections, and spiro locants as derived from the provided metadata.
   (a) **Define and label atoms/rings.** Explicitly assign ring labels and atom labels to each ring in the ring system (e.g., "Ring A: C1–C6 clockwise starting at the junction with ring B"; include heteroatoms like O1/N5 as needed). For complex fused or multi-ring systems, it is recommended to align the atom labeling by the structure metadata to maintain chemical accuracy. If there are multiple distinct fused systems, assign separate, non-overlapping label sets to avoid confusion.
   (b) **Describe internal ring features.** State ring size, aromaticity/saturation, heteroatoms, and bonding patterns (alternation, single/double).
   (c) **Explain how rings are connected.** Specify **exact shared atoms or edges** (e.g., "Ring B shares the C5–C6 edge with ring A"). Describe **fusion geometry** (linear, angular, bridged, spiro).

   Required verification (before finalizing your description):

   - **Label consistency:** Each label you introduced is used consistently; no duplicate or skipped numbering within a ring.
   - **Ring labeling check:** Verify that the atom labeling and numbering sequence are correct, and that the shared atoms or edges described for each fusion correspond accurately to your own ring definitions and orientations. The metadata should be treated as the **gold standard** for validating both labeling and fusion topology.
   - **Orientation sanity-check:** The shared atoms/edge and the named orientation (linear vs angular; inner vs outer edge) match the topology. Do not accidentally mirror or rotate the fusion.
   - **Cross-consistency check:** Ensure that substituent positions and stereochemical designations correspond correctly to the atom labels and numbering scheme you defined for each ring.
7. **Stereochemistry**
   Include stereochemical information such as (R/S) or (E/Z) when available, and describe how these configurations relate to surrounding atoms or bonds.
8. **Rational Use of Metadata**

   - Treat the metadata as **accurate structural evidence**, but express its meaning in **your own words**.
   - The metadata is not shown to the reader, so do not include or reference any of its raw contents directly.
   - Only mention atoms, labels, or locants (e.g., "C9," "N5") **after you have introduced them** in your own description.
   - When metadata provides **partial or shorthand notations**, infer and expand these to their complete, chemically correct forms.

9. **Use of Chemical Shorthand**
   - **Avoid full structural formulas** written as continuous symbolic notations.
   - Short fragments like "–OH," "–CH3," or "–NH2" are acceptable when helpful.
10. **Balance and Readability**
    Aim for a **balanced level of detail**.
11. **Descriptive Diversity**
    Use varied styles and sentence structures.
12. **Do Not Include**
    - The full IUPAC name, SMILES string, or XML tags verbatim.
    - Long symbolic chemical formulas for the entire molecule.
    - Brand names, trivial comments, or unrelated metadata.
    - Unintroduced atom labels or locants.
13. **Non-hydrogen atom count**
    - After completing the description, report the **total number of non-hydrogen atoms**.
    - Do **not** use the SMILES to perform this count.
    - Do **not** include the counting process in your description.

Output Format:

```
<description>
[Concise, varied, and chemically precise structural description]
</description>
<non_hydrogen_atom_count>
[integer]
</non_hydrogen_atom_count>
```

This section of the prompt specifies semantics for *fused*, *spiro*, and *bridged* ring systems, which are automatically injected when required. Detailed prompt specifications for these semantics are provided in prompts 4, 5, and 6, respectively.

Inputs:

- **IUPAC Name:** {IUPAC}
- **SMILES String:** {SMILES}
- **Molecular Metadata (XML Hierarchy):**

  {XML_METADATA}

Prompt 3: Molecular Description Generation from SMILES and IUPAC Names

Task:

You are provided with the **IUPAC name** and **SMILES string**, generate a **detailed and accurate structural description** that would allow a person with **basic organic chemistry knowledge** to reconstruct the exact molecule using only your structural description text. The description should be **natural**, **diverse**, and **chemically precise**, clearly conveying the molecular skeleton, substituents, and stereochemistry.

In addition, after completing the description, report the total number of non-hydrogen atoms implied by your description, computed **only from the information explicitly stated in the description itself**.

Guidelines:

1. **Purpose and Independence**
   - The description must be **self-contained** and **sufficient for reconstruction**.
   - Assume the reader only has your final text—they will **not see** the SMILES or IUPAC name.
   - You may use all input data internally to reason about the molecule, but the final description must read as a stand-alone, human-readable explanation.

2. **Freedom of Description**
   You may begin from any perspective—the main skeleton, a key ring system, or an important substituent. Combine information freely from the IUPAC name and SMILES to capture the complete structure.
3. **Backbone and Connectivity**
   Describe how rings, chains, and substituents are connected. Indicate branching positions, linkages, and overall topology so that the structure can be reconstructed accurately.
4. **Functional Groups and Substituents**
   Identify all key functional groups and substituents. Specify their type (e.g., hydroxyl, amine, halogen, carbonyl), location, and bonding pattern relative to the molecular framework.
5. **Simple or Isolated Rings**
   For common rings (benzene, pyridine, cyclohexane), you may name them directly or briefly describe their composition and bonding.
6. **Fused, Bridged, or Spiro Ring Systems — explicit, structured, and verified**
   It is **strongly recommended to follow the guidance below** when describing complex ring topologies:
   (a) **Define and label atoms/rings.** Explicitly assign ring labels and atom labels to each ring in the ring system (e.g., "Ring A: C1–C6 clockwise starting at the junction with ring B"; include heteroatoms like O1/N5 as needed). If there are multiple distinct fused systems, assign separate, non-overlapping label sets to avoid confusion.
   (b) **Describe internal ring features.** State ring size, aromaticity/saturation, heteroatoms, and bonding patterns (alternation, single/double).
   (c) **Explain how rings are connected.** Specify **exact shared atoms or edges** (e.g., "Ring B shares the C5–C6 edge with ring A"). Describe **fusion geometry** (linear, angular, bridged, spiro).

   Required verification (before finalizing your description):
   - **Label consistency:** Each label you introduced is used consistently; no duplicate or skipped numbering within a ring.
   - **Ring labeling check:** Verify that the atom labeling and numbering sequence are correct, and that the shared atoms or edges described for each fusion correspond accurately to your own ring definitions and orientations.
   - **Orientation sanity-check:** The shared atoms/edge and the named orientation (linear vs angular; inner vs outer edge) match the topology. Do not accidentally mirror or rotate the fusion.
   - **Cross-consistency check:** Ensure that substituent positions and stereochemical designations correspond correctly to the atom labels and numbering scheme you defined for each ring.
7. **Stereochemistry**
   Include stereochemical information such as (R/S) or (E/Z) when available, and describe how these configurations relate to surrounding atoms or bonds.
8. **Use of Chemical Shorthand**
   - **Avoid full structural formulas** written as continuous symbolic notations.
   - Short fragments like "–OH," "–CH3," or "–NH2" are acceptable when helpful.
9. **Balance and Readability**
   Aim for a **balanced level of detail**.
10. **Descriptive Diversity**
    Use varied styles and sentence structures.
11. **Do Not Include**
    - The full IUPAC name or SMILES tags verbatim.
    - Long symbolic chemical formulas for the entire molecule.
    - Brand names, trivial comments, or unrelated metadata.
    - Unintroduced atom labels or locants.
12. **Non-hydrogen atom count**
    - After completing the description, report the **total number of non-hydrogen atoms**.
    - Do **not** use the SMILES to perform this count.
    - Do **not** include the counting process in your description.

Output Format:

```
<description>
[Concise, varied, and chemically precise structural description]
</description>
<non_hydrogen_atom_count>
[integer]
</non_hydrogen_atom_count>
```

Inputs:

- **IUPAC Name:** {IUPAC}
- **SMILES String:** {SMILES}

Prompt 4: Fused Ring Semantics (XML)

A fused ring system is represented in the XML by **incrementally constructing new ring identities** from simpler rings. Each fusion step introduces a **new global labeling scheme**, which replaces all previous ones and is used for any subsequent fusion and structural references.

Core semantics

1. **Local ring definition**
   - Each ring is described by a `value` attribute containing its **SMILES representation**; in some fused ring systems, this SMILES value may be omitted.
   - A `labels` attribute assigns **atom labels** for the ring; when a SMILES value is present, the labels follow the **atom order in the SMILES**.
     - If `labels` is **explicitly provided**, those labels define the atom labeling scheme.
     - If `labels` is **omitted** or set to `numeric`, atoms are implicitly labeled from 1 to n, where n is the number of atoms in the ring, following SMILES order.
   - Atom indices **start from 1**.
2. **Fusion via `originalLabels`**
   - `originalLabels` maps **each atom of the newly fused system** to atom indices in the component rings.
   - Each entry corresponds to **one atom in the fused system**.
   - Multiple indices in an entry indicate a **fusion point**.
   - A blank position indicates that the fused-system atom does not belong to that ring.
3. **Label propagation**
   - After fusion, the fused system receives a new `labels` list.
   - This labeling scheme **replaces all previous local labels**.
   - All subsequent structural references—including further fusions, bridged connections, spiro connections, substituent attachment positions, and stereochemical descriptors—**must reference this new labeling scheme**.

Worked Example: indeno[5,6-b]furan

```
<group type="ring" subType="fusedRing" value="indeno[5,6-b]furan">
  <fusedChildRing
    type="ring"
    subType="ring"
    value="o1cccc1"
    labels="1/2/3/4/5">
  furan
  </fusedChildRing>
  <fusedChildRing
    type="ring"
    subType="fusionRing"
    value="[cH2]1ccc2ccccc12"
    labels="1/2/3/3a/4/5/6/7/7a"
    fusedRing1="[cH2]1cccc1"
    fusedRing2="c1ccccc1"
    originalLabels="(1,)/(2,)/(3,)/(4,1)/(,2)/(,3)/(,4)/(,5)/
    ↪  (5,6)">
  indeno
  </fusedChildRing>
  <fusedRingLabels
    labels="1/2/3/3a/4/4a/5/6/7/7a/8/8a"
    originalLabels="(1,)/(5,)/(4,)/(3,6)/(,7)/(,7a)/(,1)/(,2)/
    ↪  (,3)/(,3a)/(,4)/(2,5)">
  </fusedRingLabels>
</group>
```

Step-by-step interpretation

Step 1: Ring A (furan)

- SMILES: `o1cccc1`
- Labels: `1/2/3/4/5`
- Label assignment follows SMILES order:
  - O → 1
  - Carbons → 2–5

Step 2: Ring B (indeno)

Ring B is formed by fusing two sub-rings:

- `fusedRing1`: `[cH2]1cccc1` → atoms 1–5
- `fusedRing2`: `c1ccccc1` → atoms 1–6

Fusion is defined by:

```
originalLabels = (1,)/(2,)/(3,)/(4,1)/(,2)/(,3)/(,4)/(,5)/(5,6)
```

Fusion points:

- atom 4 of `fusedRing1` ↔ atom 1 of `fusedRing2`
- atom 5 of `fusedRing1` ↔ atom 6 of `fusedRing2`

After fusion, Ring B receives a **new labeling scheme**:

```
labels = 1/2/3/3a/4/5/6/7/7a
```

From this point onward, **indeno is treated as a single ring**.

Step 3: Fuse Ring A with Ring B

The final fused system (indeno[5,6-b]furan) is created by fusing:

- Ring A (labels `1/2/3/4/5`)
- Ring B (labels `1/2/3/3a/4/5/6/7/7a`)

The final system receives a **new global labeling scheme**:

```
labels = 1/2/3/3a/4/4a/5/6/7/7a/8/8a
```

Fusion is defined by:

```
(1,)/(5,)/(4,)/(3,6)/(,7)/(,7a)/(,1)/(,2)/(,3)/(,3a)/(,4)/(2,5)
```

Fusion points:

- atom 3 of Ring A ↔ atom 6 of Ring B
- atom 2 of Ring A ↔ atom 5 of Ring B

Prompt 5: Bridged Ring Semantics (XML)

A bridged ring system is represented by defining a **parent ring system** and a **bridge fragment** that connects two atoms of the parent ring.

For a bridged ring system, the XML normally provides:

- A **parent ring system**, defined by its `value` (SMILES, when provided) and `labels`. The parent ring may itself be a fused ring system, following the fused-ring semantics.
- A **bridge fragment**, defined by its own `value` (SMILES) and `labels`.
- A **`bridgeLocants`** attribute, which specifies the **two atom labels of the parent ring** that are connected by the bridge.

The `bridgeLocants` are ordered and define the **direction along the bridge**. The labels of the bridge fragment are assigned **from the first locant to the second locant**, following the order of the bridge fragment's SMILES.

Worked Example: 4a,8a-propanoquinoline

```
<group type="ring" subType="bridgeSystem" value="propanoquinoline">
  <bridgeParent
    type="ring"
    subType="ring"
    value="n1cccc2ccccc12"
    labels="1/2/3/4/4a/5/6/7/8/8a"
    fusedRing1="n1ccccc1"
    fusedRing2="c1ccccc1"
    originalLabels="(1,)/(2,)/(3,)/(4,)/(5,1)/(,2)/(,3)/(,4)/(,5)/
    ↪  (6,6)">
  quinoline
  </bridgeParent>
  <bridgeChild
    type="chain"
    subType="alkaneStem"
    value="-CCC-"
    labels="11/10/9"
    usableAsAJoiner="yes"
    bridgeLocants="4a,8a">
  prop
  </bridgeChild>
</group>
```

Interpretation:

- Parent ring labels: `1/2/3/4/4a/5/6/7/8/8a`
- Bridge locants: `4a,8a`
- Bridge labels: `11/10/9`

The bridge is incorporated directionally as: **4a – 11 – 10 – 9 – 8a**

The resulting structure uses a **single combined labeling scheme**, consisting of the parent-ring labels extended by the bridge labels. All subsequent references (including further fusions, substituents, and stereochemistry) must use this combined labeling scheme.

Prompt 6: Spiro Ring Semantics (XML)

For a spiro ring system, the XML normally provides:

- One or more `spiroSystemComponent` entries, each describing a **complete ring system**.
  - Each component has a `value` (SMILES) and a `labels` attribute.
  - If `labels` is **explicitly provided**, those labels are used; if `labels` is **omitted** or set to `numeric`, atoms are implicitly labeled from 1 to n, following the **SMILES atom order**.
  - A component may itself be a fused ring system, following the fused-ring semantics.
- A `spiroLocant`, which specifies the **spiro atom in each ring system**.

Each ring system **retains its own labeling scheme**. To distinguish atoms belonging to different ring systems, **prime notation** ($'$, $''$, ...) is used.

The locants in `spiroLocant` are listed in the same order as the `spiroSystemComponent` entries and identify the single atom in each ring system that represents the same physical atom.

Worked Example: spiro[cyclopentane-1,1′-indene]

```
<group type="spiro system" subType="Polycyclic"
↪  value="spiro,pent,inden">
  <spiroSystemComponent
    type="ring"
    subType="alkaneStem"
    value="C1CCCC1"
    labels="numeric">
  pent
  </spiroSystemComponent>
  <spiroLocant>1,1'</spiroLocant>
  <spiroSystemComponent
    type="ring"
    subType="ring"
    value="[cH2]1ccc2ccccc12"
    labels="1/2/3/3a/4/5/6/7/7a"
    fusedRing1="[cH2]1cccc1"
    fusedRing2="c1ccccc1"
    originalLabels="(1,)/(2,)/(3,)/(4,1)/(,2)/(,3)/(,4)/(,5)/
    ↪  (5,6)">
  inden
  </spiroSystemComponent>
</group>
```

Interpretation:

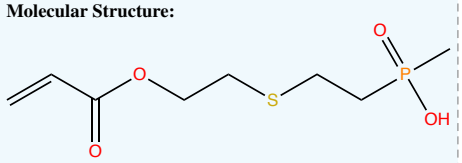
- **Cyclopentane** uses labels `1–5`; the spiro atom is **1**.
- **Indene** uses its own labeling scheme; the spiro atom is **1′**.
- Atoms **1** and **1′** refer to the **same physical atom**.

All other atoms are referenced using their **component-specific labels with primes**, consistent with the XML.

E.3 Representative Samples

We present representative molecule–description pairs from each generation difficulty level (easy, medium, and hard), with two examples per category. For each level, we select one molecule with a relatively simple structure and one with a more complex structure. For the structurally simpler examples, we additionally include the structured XML metadata constructed by our method.

Molecular Structure:



Generated Description:

The molecule is a substituted phosphinic acid built around a tetra-coordinate phosphorus atom. At the phosphorus center, place one oxo group (a P=O double bond) and one hydroxyl group (a single P–O–H bond), giving the phosphinic acid functionality. The remaining two substituents on phosphorus are: (1) a methyl group (–CH3) directly bonded to phosphorus, and (2) a two-carbon alkyl chain (–CH2–CH2–) also directly bonded to phosphorus.

From the end of that phosphorus-bound ethyl chain, connect a sulfur atom as a thioether (so the chain continues as –CH2–CH2–S–). From sulfur, attach another ethyl linker (–CH2–CH2–) that terminates at an oxygen atom (–O–). That oxygen is the single-bond (alkoxy) oxygen of an ester: it is bonded to the carbonyl carbon of a prop-2-enoyl (acryloyl) group. Complete the acyl fragment as an α,β-unsaturated carbonyl: the carbonyl carbon is double-bonded to one oxygen, and single-bonded to a vinyl group so that the chain is –C(=O)–CH=CH2 (terminal alkene).

```
<molecule name="methyl-[2-(2-prop-2-enoyloxyethylsulfanyl)ethyl]phosphinic acid">
 <wordRule wordRule="simple" type="full"
 ↪  value="methyl-[2-(2-prop-2-enoyloxyethylsulfanyl)ethyl]phosphinic acid">
  <word type="full" value="methyl-[2-(2-prop-2-enoyloxyethylsulfanyl)ethyl]phosphinic acid">
   <substituent>
    <group type="chain" subType="alkaneStem" value="C" labels="numeric"
    ↪  usableAsAJoiner="yes" resolved="yes">meth</group>
    <suffix type="inline" value="yl">yl</suffix>
    <hyphen value="-">-</hyphen>
   </substituent>
   <bracket>
    <bracket locant="2">
     <bracket type="implicit">
      <bracket type="implicit" locant="2">
       <substituent>
        <group type="chain" subType="alkaneStem" value="CCC" labels="numeric"
        ↪  usableAsAJoiner="yes" subsequentUnsemanticToken="-" defaultInID="3"
        ↪  resolved="yes">
         <unsaturator value="2" locant="2">en</unsaturator>
        </group>
        <suffix type="inline" value="oyl">oyl</suffix>
       </substituent>
       <substituent>
        <group type="substituent" subType="multiRadicalSubstituent" value="O" outIDs="1,1"
        ↪  usableAsAJoiner="yes" acceptsAdditiveBonds="yes" isAMultiRadical="2"
        ↪  resolved="yes">oxy</group>
       </substituent>
      </bracket>
      <substituent>
       <group type="chain" subType="alkaneStem" value="CC" labels="numeric"
       ↪  usableAsAJoiner="yes" defaultInID="2" resolved="yes">eth</group>
       <suffix type="inline" value="yl">yl</suffix>
      </substituent>
     </bracket>
     <substituent>
      <group type="chain" subType="heteroStem" value="S" labels="numeric"
      ↪  usableAsAJoiner="yes" resolved="yes">sulf</group>
      <suffix type="inline" value="yl">yl</suffix>
     </substituent>
    </bracket>
    <substituent>
     <group type="chain" subType="alkaneStem" value="CC" labels="numeric"
     ↪  usableAsAJoiner="yes" defaultInID="2" resolved="yes">eth</group>
     <suffix type="inline" value="yl">yl</suffix>
    </substituent>
   </bracket>
   <root>
    <group type="nonCarboxylicAcid" subType="ylForAcyl" value="[PH2](O)=O"
    ↪  usableAsAJoiner="yes">phosphin</group>
    <suffix type="root" subType="terminal" value="ic">ic acid</suffix>
   </root>
  </word>
 </wordRule>
</molecule>
```

(a) Easy category example 1 (simple structure): *methyl-[2-(2-prop-2-enoyloxyethylsulfanyl)ethyl]phosphinic acid*

Generated Description:

Begin with a 2-hydroxyacetic acid (glycolic acid) unit: a carboxylic acid group (–C(=O)OH) whose adjacent (α) carbon bears a hydroxyl group (–OH) and is stereogenic with **S** configuration. This stereogenic α carbon is bonded directly (by a C–C single bond) to a large macrocyclic substituent described next.

Construct a **28-membered macrocycle** and label the ring atoms consecutively **1→28** as follows (then close the ring by bonding atom 28 back to atom 1):

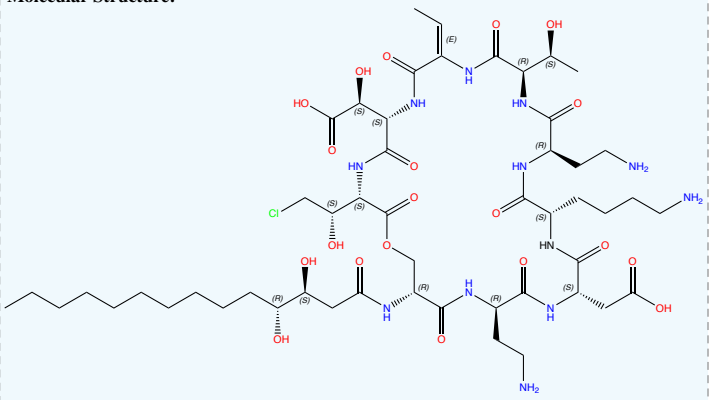
- **Atom 1**: an **oxygen** (an ether/ester oxygen) in the ring.
- **Atom 2**: a **carbonyl carbon** attached to atom 1; atom 2 is **C(=O)** (the carbonyl oxygen is outside the ring), and atom 2 is also bonded to atom 3. (Thus, atoms 1–2–3 form an **ester** linkage: O–C(=O)–C.)
- **Atom 3**: a stereogenic **carbon** with **S** configuration, bonded to atom 2 and to atom 4.
- **Atom 4**: an **amide nitrogen** (–NH–) bonded to atom 3 and to atom 5.
- **Atom 5**: a **carbonyl carbon** (C(=O)) bonded to atom 4 and atom 6.
- **Atom 6**: a stereogenic **carbon** with **S** configuration, bonded to atom 5 and atom 7; this is the **attachment point to the glycolic acid α carbon** described at the beginning (the glycolic acid stereocenter is directly bonded to ring atom 6).
- **Atom 7**: an **amide nitrogen** (–NH–) bonded to atom 6 and atom 8.
- **Atom 8**: a **carbonyl carbon** (C(=O)) bonded to atom 7 and atom 9.
- **Atom 9**: a **vinylic ring carbon** bonded to atom 8 and atom 10, and double-bonded to an **exocyclic ethylidene group**: atom 9 is double-bonded to an external carbon that bears one hydrogen and one **methyl** substituent (i.e., =CH–CH_3). The C9=C (exocyclic) double bond has **E** geometry.
- **Atom 10**: an **amide nitrogen** (–NH–) bonded to atom 9 and atom 11.
- **Atom 11**: a **carbonyl carbon** (C(=O)) bonded to atom 10 and atom 12.
- **Atom 12**: a stereogenic **carbon** with **R** configuration, bonded to atom 11 and atom 13.
- **Atom 13**: an **amide nitrogen** (–NH–) bonded to atom 12 and atom 14.
- **Atom 14**: a **carbonyl carbon** (C(=O)) bonded to atom 13 and atom 15.
- **Atom 15**: a stereogenic **carbon** with **R** configuration, bonded to atom 14 and atom 16.
- **Atom 16**: an **amide nitrogen** (–NH–) bonded to atom 15 and atom 17.
- **Atom 17**: a **carbonyl carbon** (C(=O)) bonded to atom 16 and atom 18.
- **Atom 18**: a stereogenic **carbon** with **S** configuration, bonded to atom 17 and atom 19.
- **Atom 19**: an **amide nitrogen** (–NH–) bonded to atom 18 and atom 20.
- **Atom 20**: a **carbonyl carbon** (C(=O)) bonded to atom 19 and atom 21.
- **Atom 21**: a stereogenic **carbon** with **S** configuration, bonded to atom 20 and atom 22.
- **Atom 22**: an **amide nitrogen** (–NH–) bonded to atom 21 and atom 23.
- **Atom 23**: a **carbonyl carbon** (C(=O)) bonded to atom 22 and atom 24.
- **Atom 24**: a stereogenic **carbon** with **R** configuration, bonded to atom 23 and atom 25.
- **Atom 25**: an **amide nitrogen** (–NH–) bonded to atom 24 and atom 26.
- **Atom 26**: a **carbonyl carbon** (C(=O)) bonded to atom 25 and atom 27.
- **Atom 27**: a stereogenic **carbon** with **R** configuration, bonded to atom 26 and atom 28.
- **Atom 28**: a **methylene carbon** (–CH_2–) bonded to atom 27 and back to atom 1 to close the macrocycle.

Add the substituents on the ring carbons (all other ring nitrogens are the secondary amide –NH– described above):

1. **At ring atom 3** (the S stereocenter): attach a **2-chloro-1-hydroxyethyl** group in which the carbon directly attached to ring atom 3 is itself stereogenic with **S** configuration and bears –OH; that carbon is bonded to a terminal **–CH_2Cl** group.
2. **At ring atom 12** (R): attach a **1-hydroxyethyl** group where the carbon attached to ring atom 12 is stereogenic with **S** configuration and bears –OH and a **methyl** group (i.e., –CH(OH)–CH_3).
3. **At ring atom 15** (R): attach a **2-aminoethyl** substituent, –CH_2–CH_2–NH_2.
4. **At ring atom 18** (S): attach a **4-aminobutyl** substituent, –CH_2–CH_2–CH_2–CH_2–NH_2.
5. **At ring atom 21** (S): attach a **carboxymethyl (acetic acid)** substituent, –CH_2–C(=O)OH.
6. **At ring atom 24** (R): attach a second **2-aminoethyl** substituent, –CH_2–CH_2–NH_2.
7. **At ring atom 27** (R): attach an **amide side chain** via an –NH– group directly bonded to ring atom 27 (so ring atom 27 bears a substituent –NH–C(=O)–…). This amide nitrogen is acylated by a **tetradecanoyl** chain (14 carbons counting the carbonyl carbon) that is linear and unbranched and contains two hydroxyl groups at the 3- and 4-positions from the carbonyl: –C(=O)–CH_2–CH(OH)–CH(OH)–$(CH_2)_9$–CH_3, with the hydroxyl-bearing carbons having **3S** and **4R** configurations (counting from that side-chain carbonyl carbon as position 1).

This completes the full connectivity and stereochemistry (including the **E** ethylidene double bond at ring atom 9 and the stated absolute configurations at every stereocenter).

Molecular Structure:



(b) Easy category example 2 (complex structure): *(2S)-2-[(3S,6S,9E,12R,15R,18S,21S,24R,27R)-18-(4-azanylbutyl)-15,24-bis(2-azanylethyl)-27-[[(3S,4R)-3,4-bis(oxidanyl)tetradecanoyl]amino]-3-[(1S)-2-chloranyl-1-oxidanyl-ethyl]-9-ethylidene-21-(2-hydroxy-2-oxoethyl)-12-[(1S)-1-oxidanylethyl]-2,5,8,11,14,17,20,23,26-nonakis(oxidanylidene)-1-oxa-4,7,10,13,16,19,22,25-octazacyclooctacos-6-yl]-2-oxidanyl-ethanoic acid*

Generated Description:

The molecule is a diaryl ketone (a "benzophenone-type" structure): a single carbonyl group (C=O) whose carbon is bonded to two aromatic substituents.

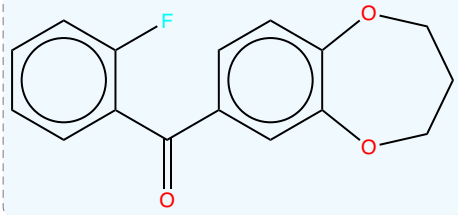
First build the fused bicyclic aryl substituent (a benzene fused to a 7-membered dioxacycle). Define it with these atom labels:

- Ring A (benzene): six aromatic carbons arranged as C5a–C6–C7–C8–C9–C9a–(back to C5a).
- Ring B (seven-membered heterocycle): fuse this ring to Ring A by sharing the C5a–C9a bond. Ring B is C9a–O1–C2–C3–C4–O5–C5a–(back to C9a), where O1 and O5 are ether oxygens and C2, C3, and C4 are three consecutive saturated methylene carbons (–CH2– units).

At aromatic carbon C7 of Ring A, attach the carbonyl carbon of the ketone (the carbonyl oxygen is double-bonded to this carbonyl carbon).

The other substituent on the carbonyl carbon is a phenyl ring. Label the phenyl carbon bonded directly to the carbonyl as C1'; place a fluorine atom on the adjacent ring carbon C2' (i.e., ortho to the carbonyl attachment). The remaining four phenyl positions are unsubstituted.

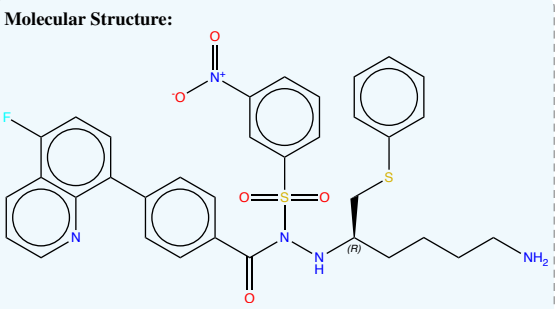
Molecular Structure:




```
<molecule name="3,4-dihydro-2H-1,5-benzodioxepin-7-yl-(2-fluorophenyl)methanone">
 <wordRule wordRule="simple" type="full"
 ↪  value="3,4-dihydro-2H-1,5-benzodioxepin-7-yl-(2-fluorophenyl)methanone">
  <word type="full" value="3,4-dihydro-2H-1,5-benzodioxepin-7-yl-(2-fluorophenyl)methanone">
   <substituent>
    <group type="ring" subType="fusedRing" value="benzooxoxepin" resolved="yes">
     <fusedChildRing type="ring" subType="fusionRing" value="c1ccccc1"
     ↪  labels="1/2/3/4/5/6">benzo</fusedChildRing>
     <fusedChildRing type="ring" subType="hantzschWidman" value="c1cccccc1" labels="numeric"
     ↪  subsequentUnsemanticToken="-">
      <heteroatom value="O" multiplied="multiplied" resolved="yes" locant="1">ox</heteroatom>
      <heteroatom value="O" multiplied="multiplied" resolved="yes" locant="5">ox</heteroatom>
     </fusedChildRing>
     <fusedRingLabels labels="1/2/3/4/5/5a/6/7/8/9/9a" originalLabels="(, 1)/(, 2)/(, 3)/(,
     ↪  4)/(, 5)/(1, 6)/(6, )/(5, )/(4, )/(3, )/(2, 7)"></fusedRingLabels>
     <indicatedHydrogen locant="2"></indicatedHydrogen>
     <hydro value="hydro" subsequentUnsemanticToken="-" multiplied="multiplied"
     ↪  locant="4">hydro</hydro>
     <hydro value="hydro" subsequentUnsemanticToken="-" multiplied="multiplied"
     ↪  locant="3">hydro</hydro>
    </group>
    <suffix type="inline" value="yl" locant="7">yl</suffix>
    <hyphen value="-">-</hyphen>
   </substituent>
   <bracket>
    <substituent locant="2">
     <group type="substituent" subType="halideOrPseudoHalide" value="-[F|1]"
     ↪  resolved="yes">fluoro</group>
    </substituent>
    <substituent>
     <group type="ring" subType="arylSubstituent" value="-c1ccccc1"
     ↪  labels="1/2,ortho/3,meta/4,para/5/6" resolved="yes">phenyl</group>
    </substituent>
   </bracket>
   <root>
   <group type="chain" subType="alkaneStem" value="C" labels="numeric"
   ↪  usableAsAJoiner="yes">meth</group>
   <suffix type="root" value="one">one</suffix>
   </root>
  </word>
 </wordRule>
</molecule>
```

(c) Medium category example 1 (simple structure): *3,4-dihydro-2H-1,5-benzodioxepin-7-yl-(2-fluorophenyl)methanone*

Molecular Structure:



Generated Description:

Start from an acyl hydrazide core derived from a para-substituted benzoic acid.

1. Define Ring A as a benzene ring containing 6 carbon atoms, labeled A1–A6 consecutively. At A1, attach a carbonyl group so that the carbonyl carbon (one carbon atom) is directly bonded to A1, is double-bonded to one oxygen atom, and is single-bonded to an amide nitrogen N2.
2. Hydrazide linkage: connect N2 by a single bond to a second nitrogen N1 (so the sequence is carbonyl carbon–N2–N1). N2 bears no hydrogen because it is substituted by the carbonyl carbon, by N1, and by a sulfonyl group (next step).
3. Sulfonamide substituent on N2: bond N2 to a sulfonyl sulfur atom S_sulf. This sulfur is double-bonded to two oxygen atoms and single-bonded to Ring B (a phenyl ring). Define Ring B as a benzene ring of 6 carbon atoms labeled B1–B6, with B1 bonded to S_sulf. Place a nitro group (–NO2, i.e., one nitrogen atom and two oxygen atoms) on B3, which is the meta position relative to B1.
4. Substituent on N1: bond N1 to a stereogenic carbon X2. Build a straight six-carbon chain X1–X2–X3–X4–X5–X6 (six carbon atoms total), where X2 is the chiral center. The segment X3–X4–X5–X6 consists of four consecutive methylene carbons, and X6 carries a terminal primary amino group (one nitrogen atom, –NH2).
5. Thioether at X1: X1 is a methylene carbon attached to X2 and also to a thioether sulfur atom S_thio. Bond S_thio to Ring C, an unsubstituted phenyl ring containing 6 carbon atoms.
6. Absolute configuration: X2 has the R configuration. Using CIP priorities at X2 (N1 has highest priority; X1 next because it leads to sulfur; X3 next; hydrogen lowest), orient the X2–H bond away from the viewer; the order N1 → X1 → X3 then traces a clockwise path.
7. Para quinoline substituent on Ring A: at A4 (para to A1), attach a quinoline group through a single bond from A4 to atom Q8 of the quinoline. Construct the quinoline as an aromatic fused bicyclic system with 10 ring atoms total (9 carbons and 1 ring nitrogen). Label the pyridine-like ring as Q1 (the ring nitrogen), then Q2–Q3–Q4–Q4a–Q8a returning to Q1. Fuse to it a benzene ring that shares the Q4a–Q8a bond and follows Q4a–Q5–Q6–Q7–Q8–Q8a–Q4a. Place a fluorine atom on Q5.

(d) Medium category example 2 (complex structure): *N'-[(2R)-6-azanyl-1-phenylsulfanyl-hexan-2-yl]-4-(5-fluoranylquinolin-8-yl)-N-(3-nitrophenyl)sulfonyl-benzohydrazide*

Generated Description:

Build a fully conjugated, all-aromatic molecule that contains two substituted benzothiadiazole units connected by a thiophene–difluorophenyl–thiophene linker, and with the "inner" benzothiadiazole additionally attached to a tricyclic sulfur-containing fused core that bears two identical fluoro/hexyl thiophene substituents.

1. **Terminal benzothiadiazole unit (Unit A)**
   - Make an aromatic "benzothiadiazole" fused bicyclic system by fusing:
     - a benzene ring (6 carbons) and
     - a 1,2,5-thiadiazole ring (a 5-member aromatic ring whose atom sequence around the ring is N–S–N–C–C),

     with the fusion occurring through the two carbon atoms of the thiadiazole (so the fused system contains 6 ring carbons, 2 ring nitrogens, and 1 ring sulfur; 9 non-hydrogen atoms in the fused core).
   - On the benzene portion of this fused system, there are four consecutive carbons that are not part of the fusion. Label these four benzene carbons A1–A4 in order around that benzene ring.
     - A1 is bonded to a thiophene ring (thiophene Tα, described below).
     - A2 and A3 each bear a 2-ethylhexyloxy substituent (an ether: ring–O–(2-ethylhexyl), where the 2-ethylhexyl group is the 8-carbon chain O–CH2–CH(–CH2CH3)–CH2–CH2–CH2–CH3).
     - A4 bears a methyl group (–CH3).
2. **Linker from Unit A to the second benzothiadiazole**
   - Thiophene Tα: an aromatic thiophene ring (5-member ring with 1 sulfur and 4 carbons). Use the usual thiophene positions where the two "α" carbons are the two carbons adjacent to sulfur. Connect Tα through its two α carbons: one α carbon is bonded to A1 of Unit A, and the opposite α carbon is bonded to a difluorophenyl ring (Unit P) described next.
   - Unit P (difluorophenyl): a benzene ring (6 carbons). Label the ring carbons P1–P6 consecutively around the ring.
     - Bond P1 to thiophene Tα.
     - Bond the para carbon P4 to a second thiophene ring (Tβ, next bullet).
     - Put fluorine atoms on P2 and P5 (so each fluorine sits on a carbon adjacent to one of the two para substituent positions).
   - Thiophene Tβ: another aromatic thiophene ring (again 5 atoms: 1 sulfur, 4 carbons), substituted at its two α carbons. Bond one α carbon of Tβ to P4 of Unit P, and bond the opposite α carbon of Tβ to the second benzothiadiazole unit (Unit B) described below.
3. **Second benzothiadiazole unit (Unit B)**
   - Create a second benzothiadiazole fused system with the same ring composition as Unit A (again 6 ring carbons, 2 ring nitrogens, 1 ring sulfur).
   - On the benzene portion of Unit B, label the four non-fusion benzene carbons B1–B4 in order around that benzene ring.
     - B1 is bonded to the tricyclic fused core (Core X, defined next) by a direct C–C bond.
     - B2 and B3 each bear a 2-ethylhexyloxy group of the same structure defined above (ring–O–CH2–CH(–CH2CH3)–CH2–CH2–CH2–CH3).
     - B4 is bonded by a C–C bond to thiophene Tβ (from step 2).
4. **Tricyclic fused sulfur-containing core (Core X) attached to Unit B**

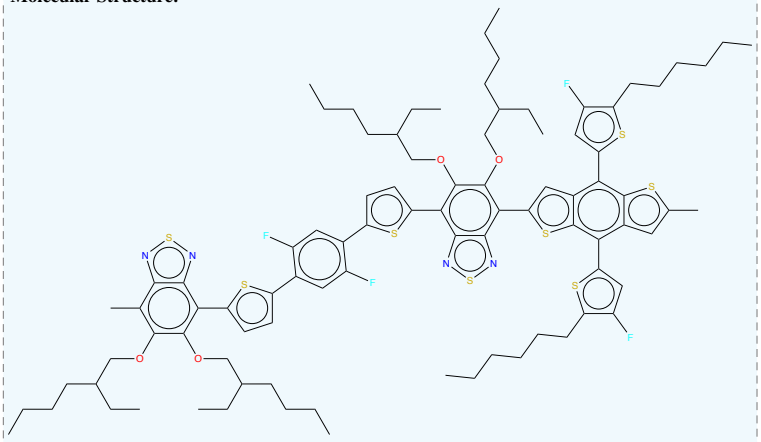
   Define Core X explicitly as a 12-atom fused aromatic system with atom labels 1, 2, 3, 3a, 4, 4a, 5, 6, 7, 7a, 8, 8a:
   - Atoms 1 and 5 are sulfur; all other labeled atoms in Core X are aromatic carbons.
   - Ring L (a thiophene): 1–2–3–3a–8a–1.
   - Ring M (a benzene): 7a–8–8a–3a–4–4a–7a.
   - Ring R (a thiophene): 5–6–7–7a–4a–5.

   (So Ring L and Ring M share the 3a–8a edge, and Ring R and Ring M share the 4a–7a edge.)

   Substitution on Core X:
   - Bond B1 of Unit B to atom 2 of Core X (single C–C bond).
   - Put a methyl group (–CH3) on atom 6 of Core X.
   - Attach two identical substituted thiophene rings (step 5) to Core X by single C–C bonds at atoms 4 and 8.
5. **The two identical peripheral thiophene substituents on Core X (one on atom 4, one on atom 8)**
   - Each substituent is an aromatic thiophene ring (5-member ring with 1 sulfur and 4 carbons). Number within each such substituent thiophene as: position 1 = sulfur; position 2 = the carbon bonded to Core X; positions 3 and 4 = the two β carbons; position 5 = the other α carbon.
   - Place a fluorine atom on position 4 of this thiophene.
   - Place an n-hexyl substituent (a straight six-carbon chain, –CH2–CH2–CH2–CH2–CH2–CH3) on position 5 of this thiophene, attached via the chain's first carbon to the thiophene.

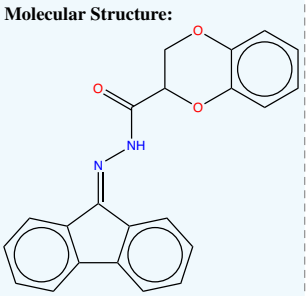
No stereochemical configuration is specified for the chiral center at the 2-position of each 2-ethylhexyl group.

Molecular Structure:



(e) Hard category example 1 (complex structure): *4-[5-[4-[5-[4-[4,8-bis(4-fluoranyl-5-hexyl-thiophen-2-yl)-6-methyl-thieno[2,3-f][1]benzothiol-2-yl]-5,6-bis(2-ethylhexoxy)-2,1,3-benzothiadiazol-7-yl]thiophen-2-yl]-2,5-bis(fluoranyl)phenyl]thiophen-2-yl]-5,6-bis(2-ethylhexoxy)-7-methyl-2,1,3-benzothiadiazole*

Molecular Structure:



Generated Description:

Build a fused bicyclic “benzodioxane” framework: start with a benzene ring (6 aromatic carbon atoms). Select two adjacent (ortho) benzene carbons and fuse to them a second six-membered ring that contains two ether oxygens and two saturated carbons, arranged so that the path between the two oxygens is –O–CH2–CH(–)–O–. Concretely, each of the two adjacent benzene carbons is bonded to one of the ring oxygens; those two oxygens are connected to each other through a two-carbon chain in which the carbon next to one oxygen is a methylene (CH2) and the next carbon is a methine (CH). This fused bicyclic unit therefore contains 8 carbons total and 2 oxygens.

On that methine carbon of the –O–CH2–CH–O– segment, attach a carboxamide carbonyl: the ring carbon is single-bonded to a carbonyl carbon that is double-bonded to an oxygen atom, and this carbonyl carbon is also single-bonded to an amide nitrogen N1 (so the substituent begins as –C(=O)–N1H–). N1 is single-bonded to a second nitrogen N2, giving a –C(=O)–NH–N= linkage, and N2 is double-bonded to the bridge carbon of a fluorene unit.

The fluorene unit is an unsubstituted 13-carbon tricyclic system made of two benzene rings fused on opposite sides of a central five-membered ring; the “bridge” carbon is the unique carbon in that five-membered ring that connects the two aromatic halves (the carbon that is CH2 in fluorene itself). In this molecule that bridge carbon has a double bond to N2 (and single bonds to the two adjacent ring-junction carbons), and there are no other substituents on the fluorene or on the benzodioxane ring system. The substituted methine carbon in the benzodioxane portion is stereogenic, but no absolute configuration is specified.


```
<molecule name="N-(fluoren-9-ylideneamino)-2,3-dihydro-1,4-benzodioxine-3-carboxamide">
 <wordRule wordRule="simple" type="full"
 ↪  value="N-(fluoren-9-ylideneamino)-2,3-dihydro-1,4-benzodioxine-3-carboxamide">
  <word type="full"
  ↪  value="N-(fluoren-9-ylideneamino)-2,3-dihydro-1,4-benzodioxine-3-carboxamide">
   <bracket locant="N">
    <substituent>
     <group type="ring" subType="ring" value="c1cccc2c3ccccc3[cH2]c12"
     ↪  labels="1/2/3/4/4a/4b/5/6/7/8/8a/9/9a" fusedRing1="c1ccccc1"
     ↪  fusedRing2="[cH2]1cccc1" fusedRing3="c1ccccc1" originalLabels="(1,,)/(2,,)/(3,,)/(4
     ↪  ,,)/(5,3,)/(,4,1)/(,,2)/(,,3)/(,,4)/(,,5)/(,5,6)/(,1,)/(6,2,)"
     ↪  subsequentUnsemanticToken="-" resolved="yes">fluoren</group>
     <suffix type="inline" value="ylidene" locant="9">ylidene</suffix>
    </substituent>
    <substituent>
     <group type="substituent" subType="simpleSubstituent" value="-N" usableAsAJoiner="yes"
     ↪  resolved="yes">amino</group>
    </substituent>
    <hyphen value="-">-</hyphen>
   </bracket>
   <root>
    <group type="ring" subType="fusedRing" value="benzooxoxin">
     <fusedChildRing type="ring" subType="fusedRing" value="benzooxoxin"
     ↪  labels="1/2,ortho/3,meta/4,para/5/6" subsequentUnsemanticToken="e-">
     <fusedChildRing type="ring" subType="fusionRing" value="c1ccccc1"
     ↪  labels="1/2/3/4/5/6">benzo</fusedChildRing>
     <fusedChildRing type="ring" subType="hantzschWidman" value="c1ccccc1"
     ↪  labels="1/2,ortho/3,meta/4,para/5/6" subsequentUnsemanticToken="e-">
      <heteroatom value="O" multiplied="multiplied" locant="1">ox</heteroatom>
      <heteroatom value="O" multiplied="multiplied" locant="4">ox</heteroatom>
     </fusedChildRing>
     <fusedRingLabels labels="1/2/3/4/4a/5/6/7/8/8a" originalLabels="(, 4)/(, 3)/(, 2)/(,
     ↪  1)/(2, 6)/(3, )/(4, )/(5, )/(6, )/(1, 5)"></fusedRingLabels>
     </fusedChildRing>
     <hydro value="hydro" subsequentUnsemanticToken="-" multiplied="multiplied"
     ↪  locant="3">hydro</hydro>
     <hydro value="hydro" subsequentUnsemanticToken="-" multiplied="multiplied"
     ↪  locant="2">hydro</hydro>
    </group>
    <suffix type="root" subType="terminal" value="amide" suffixPrefix="C"
    ↪  locant="3">amide</suffix>
   </root>
  </word>
 </wordRule>
</molecule>
```


(f) Hard category example 2 (simple structure): *N-(fluoren-9-ylideneamino)-2,3-dihydro-1,4-benzodioxine-3-carboxamide*

Figure E.1: Representative examples for the generated dataset.