

## Appendix

<b>A Data Construction Details</b>	<b>1</b>
A.1 Data Collection	1
A.2 Dataset Composition and Filtering Strategies	1
A.3 Rationale for Task Construction	2
<b>B Experimental Details</b>	<b>3</b>
B.1 Hardware Requirements	3
B.2 Evaluation Metrics	3
B.3 Count Distribution Analysis	4
<b>C Case Study for Tasks in ChemCoTBench</b>	<b>5</b>
C.1 Case Study for Molecule Understanding	5
C.2 Case Study for Molecule Editing	6
C.3 Case Study for Molecule Optimization	6
<b>D Task Example</b>	<b>9</b>

## A Data Construction Details

In this section, we propose the detailed information during our benchmark and dataset construction process, including the data source description, dataset composition, filtering strategies, and the rationale for dataset construction. In Table. 4, we also visualize the data distribution of subtasks in ChemCoTBench.

### A.1 Data Collection

The raw molecular structures used for understanding, editing, and optimization are obtained from several published datasets, including PubChem [28], ChEMBL [10], ZINC [23], and Deep-Mol-Opt [17]. Chemical reaction data are separately collected from patent databases, including USPTO [19], Pistachio [37], and Reaxys [8]. For reaction mechanism annotation, we followed the processing pipeline described in [26].

### A.2 Dataset Composition and Filtering Strategies

**Molecular Samples (25% of Benchmark):** Although the ZINC database contains 250,000 molecules, we observed that its molecular weight distribution is relatively concentrated. To ensure diversity, we carefully selected molecules from PubChem, ChEMBL, and ZINC based on molecular weight and structural complexity. This filtering process resulted in a smaller but more representative molecular subset for our benchmark.

**Molecular Optimization Pairs (38% of Benchmark):** The Deep-Mol-Opt dataset provided 198,559 molecular pairs with property annotations. However, we excluded pairs with minimal property improvement ( $\Delta < 0.3$ ) or those containing complex polycyclic structures that might challenge LLM comprehension. The remaining high-quality pairs were retained for molecular optimization tasks.

**Chemical Reaction Samples (19% of Benchmark):** Reaction equations (including reactants, products, conditions, and catalysts) were sourced from USPTO, Pistachio, and Reaxys. To avoid redundancy, we balanced the selection across these databases by reaction type and catalyst diversity. For reaction mechanism annotation, we incorporated 275 manually curated examples from [26], which were chosen for their high quality and balanced distribution.

Table 4: **The Dataset Statistics of ChemCoTBench and its Large CoT Dataset.** We visualize the sample numbers for every subtask in ChemCoTBench. The data distribution of molecule understanding & editing, molecule optimization, and reaction prediction is nearly average.

#	Mol-Understanding			Mol-Edit			Mol-Optimization		Reaction			
	Func-Group	Scaffold	SMILES	Add	Del	Sub	Physico	Protein	Fwd	Retro	Cond	Mech
Bench mark	120	100	100	20	20	60	300	300	200	100	90	275
CoT Dataset	6400			4500			3000					

### A.3 Rationale for Task Construction

**Molecular Understanding and Editing Tasks:** Molecular understanding and editing tasks are designed as closed-ended problems with deterministic answers. Since these tasks rely on well-defined chemical properties and structures, we directly sampled molecules from PubChem, ChEMBL, and ZINC as the source data. The corresponding ground-truth answers, including molecular properties and SMILES transformations, are programmatically extracted using RDKit, ensuring accuracy and reproducibility.

**Molecular Optimization Task Design:** Unlike fixed-answer tasks, molecular optimization is inherently open-ended, where multiple valid optimization paths may exist for a given input molecule. To construct this dataset, we considered two sampling strategies:

- **Baseline Model-Generated Optimizations:** *Advantage:* Enables sampling large-scale and multi-step optimization paths for source molecules; *Limitation:* Existing models often fail to preserve scaffold consistency, a critical requirement in drug design.
- **Predefined Molecular Pairs:** *Advantage:* Ensures chemically meaningful transformations with verified property improvements; *Limitation:* limited molecule samples.

To maintain the scaffold consistency, we adopt the second strategy for our ChemCoTBench, sourcing molecular pairs from Deep-Mol-Opt [17]. We perform Murcko scaffold similarity analysis to validate scaffold consistency, confirming that the selected pairs maintain structural integrity while optimizing target properties.

**Reaction Prediction Task Design:** Reaction prediction is a cornerstone of chemical research and industrial applications. From an academic standpoint, it is fundamental to understanding chemical reactivity, discovering novel transformations, and advancing the design of new molecules. In practical applications, accurate reaction prediction accelerates drug discovery, facilitates materials science innovation, optimizes chemical manufacturing processes, and enables the automation of chemical synthesis. Our benchmark aims to evaluate LLMs’ capabilities in this multifaceted domain rigorously.

- **Forward Reaction Prediction:** This task, pivotal for academic discovery and industrial applications like drug development, evaluates an LLM’s ability to predict both major products and, uniquely in our benchmark, byproducts from given reactants and reagents. Data is sourced from 100 distinct reaction classes from Pistachio. To enhance difficulty and assess deeper reasoning, the reaction type is deliberately omitted, requiring the model to first infer the plausible reaction type and then deduce potential products, thereby providing a comprehensive understanding of reaction outcomes crucial for optimization.
- **Retrosynthesis Prediction:** Essential for planning the synthesis of novel compounds, this task assesses an LLM’s understanding of reverse chemical logic, specifically its capacity to identify strategic bond disconnections and propose valid precursor structures. We focus on single-step retrosynthesis, considering multi-step planning a more complex hybrid task, to directly evaluate core retrosynthetic reasoning. Data comprises 100 reaction classes from Pistachio, and problem formulation includes providing reagents alongside the target product to help narrow the solution space and guide the LLM towards chemically relevant disconnections.
- **Reaction Condition Prediction:** Predicting optimal reaction conditions (catalysts, solvents, reagents) is critical for synthesis success, efficiency, and selectivity. This task tests an LLM’s knowledge of

how these components influence reaction pathways. Following Gao et al. [9] for data construction from USPTO [32] (retaining reactions with at most one catalyst, two solvents, and two reagents), we uniquely model this as a SMILES sequence generation task for catalyst, solvent, and reagent prediction, offering a more rigorous challenge than simple MCQ formats by requiring specific chemical structure (In SMILES) generation.

- **Mechanism Prediction:** Understanding reaction mechanisms—the step-by-step sequence of elementary reactions—is fundamental to chemistry, providing the "why" and "how" behind transformations and enabling rational design and optimization. This task evaluates an LLM’s grasp of core mechanistic principles such as electron flow, intermediate stability, bond-making/breaking sequences, and the influence of conditions on pathways, addressing a significant gap in current LLM assessments, which often treat reactions as black boxes. Inspired by prior works [25, 26] but aiming for a more holistic probe, we introduce two subtasks: "Next Elementary Step Product Prediction," where the LLM, given a sequence of annotated elementary steps, predicts the subsequent product, testing its ability to comprehend and extrapolate mechanistic progression; and "Reaction Mechanism Selection (MCQ type)," where the LLM chooses the most plausible mechanism from several alternatives for a given reaction (reactants, conditions, reagents), assessing its capacity to discern how subtle changes in reagents or conditions dictate specific mechanistic routes, thereby evaluating both sequential understanding and discriminative judgment of mechanistic pathways.

## B Experimental Details

### B.1 Hardware Requirements

The experimental workload was supported by a dedicated GPU cluster comprising three high-performance computing nodes: an NVIDIA RTX A6000 (48GB VRAM) and an RTX 3090 (24GB VRAM) for LLM API scheduling and deployment of smaller models (1.5B/7B parameters), complemented by an NVIDIA A100 (80GB VRAM) node dedicated to large-scale LLM inference. This heterogeneous configuration achieved optimal resource allocation, with the A100’s tensor cores and high-bandwidth memory handling memory-intensive model inferences while the A6000/3090 pair efficiently managed concurrent API requests and lighter workloads. Storage requirements remained modest at approximately 1GB, encompassing benchmark datasets (SMILES strings and annotations), quantized model checkpoints, and evaluation logs, all hosted on an NVMe-backed filesystem for rapid data access.

### B.2 Evaluation Metrics

To comprehensively assess model performance, we employ the following metrics:

**Accuracy:** The proportion of correctly predicted outcomes, providing a baseline measure of overall correctness. For reaction prediction tasks (e.g., forward reaction prediction), we choose the Top-1 accuracy, which specifically means the model’s highest-ranked prediction exactly matches the true product(s).

**Mean Absolute Error:** Quantifies the average magnitude of errors in continuous predictions, offering insight into precision for regression tasks (e.g., molecular property prediction).

**Scaffold Similarity:** Measured via the Tanimoto coefficient of molecular scaffolds, this evaluates structural conservation between generated and reference molecules. Values range from 0 to 1, representing scaffolds without similarity to correct scaffolds, with higher scores indicating better preservation of core frameworks.

**Improvement:** Absolute gains in target properties, reported as: Mean improvement: Average uplift across all samples. Max/min improvement: Extreme cases highlighting model potential and limitations.

**Success Rate:** The fraction of generated molecules exceeding a predefined threshold (e.g., > 0.8 for solubility), reflecting practical utility.

**Validity:** Measures the proportion of generated SMILES strings that are syntactically correct and can be successfully parsed into a chemical structure by RDKit [30].

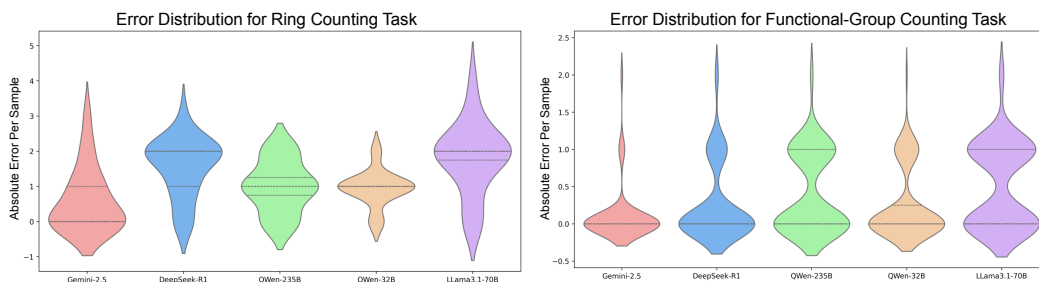


Figure 5: Error distribution analysis for ring counting and functional-group counting tasks.

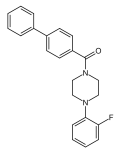
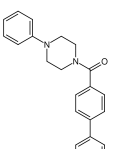
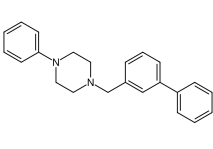
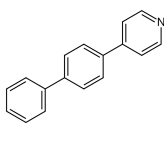
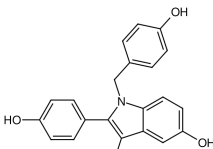
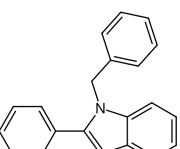
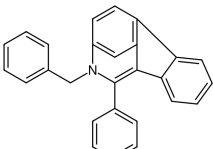
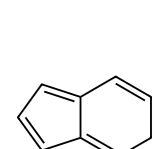
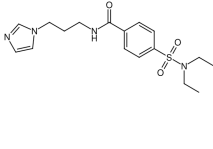
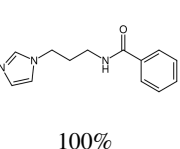
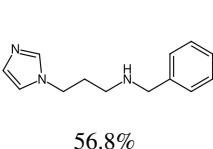
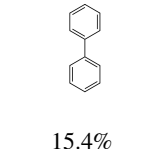
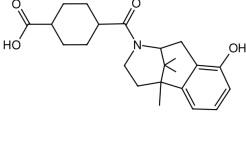
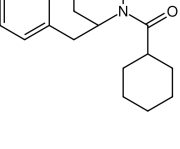
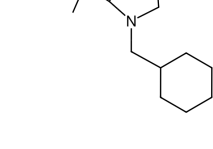

937 For the two counting tasks under molecule understanding—ring counting and functional-group  
938 counting—we evaluated model performance using the Mean Absolute Error in the main experimental  
939 section to quantify overall accuracy. To provide a more granular analysis of LLMs’ capabilities in  
940 these molecule-specific counting tasks, we further examined the error distribution across different  
941 models.

942 As illustrated in Fig. 5, the ring counting task proves significantly more challenging than the functional-  
943 group counting task. This is evident from the error distributions: For functional-group counting, the  
944 majority of errors fall within the 0.0–1.0 range, indicating relatively high accuracy. In contrast, ring  
945 counting exhibits higher errors, with most models (except Gemini-2.5-pro) showing an average MAE  
946 > 1.0. Gemini-2.5-pro stands out as the only model achieving consistently low errors in this task,  
947 suggesting superior structural reasoning capabilities. This disparity highlights the inherent difficulty  
948 of ring counting, which requires precise identification of cyclic structures—a more complex task  
949 than detecting localized functional groups. The results underscore the need for further refinement of  
950 LLMs in handling intricate molecular topologies.

## C Case Study for Tasks in ChemCoTBench

To provide a more detailed analysis of the performance of different types of LLMs across various tasks in ChemCoTBench, we supplement the quantitative findings in the Experiment section with visualizations of model outputs. In the following three subsections, we present case visualizations from distinct subtasks: molecule understanding, molecule editing, and molecule optimization.

Table 5: This is a case study for molecule understanding. We visualize the Murcko Scaffold generation task in molecule understanding because it can provide detailed information compared to number prediction tasks and correction distinguishing tasks.

Source Molecule	GT-Scaffold	Gemini-2.5-pro	Llama3.3-70B
	 100%	 41.8%	 27.8%
	 100%	 38.6%	 0.0%
	 100%	 56.8%	 15.4%
	 100%	 33.3%	 13.3%

### C.1 Case Study for Molecule Understanding

The molecule understanding task in ChemCoTBench contains three types of subtasks, including number prediction subtasks (functional-group counting and ring counting), distinguish subtasks (ring system distinguish, SMILES consistency distinguish), and scaffold generation subtask (murcko scaffold generation). To visualize the detailed molecule structure generated by different types of LLMs, we select the Murcko scaffold generation subtask as the case visualization source.

Table 5 presents four examples featuring distinct ring structures and functional groups. Through comparative analysis, we identify two key advantages of commercial LLMs over smaller open-source LLMs:

**Superior SMILES Parsing Accuracy.** Commercial LLMs (e.g., Gemini-2.5-Pro) correctly interpret molecular SMILES structures, with predicted structures closely matching the source molecules (only 1–2 bond position errors). In contrast, open-source models like LLaMA-3.1 generate structures largely inconsistent with the source molecules.

**Robust Instruction-Following for Murcko Scaffolds.** When tasked with extracting Murcko scaffolds—defined as the maximal connected framework retaining ring systems while removing non-critical functional groups—commercial LLMs adhere to the provided instructions and generate connected scaffolds. Llama-3.1, however, often outputs fragmented substructures, highlighting its limitations in instruction comprehension.

## C.2 Case Study for Molecule Editing

The molecule editing task in ChemCoTBench contains three parts: adding a target functional group to the molecule, removing a target functional group from the molecule, and substituting a functional group with a target functional group from the molecule. In Table. 6, we visualize samples from each subtask with different types of target functional groups. Two key observations emerge from the analysis:

**Functional Group Recognition Directly Impacts Task Performance.** Gemini-2.5-Pro demonstrates high precision in functional group identification, enabling accurate molecular editing. While Qwen3-235B correctly identifies functional groups, it frequently fails to execute valid molecular modifications. LLaMA-3.1 struggles with basic functional group recognition, severely limiting its task completion capability. This trend aligns with the models’ performance in the functional-group counting subtask under molecule understanding, confirming a strong correlation between recognition accuracy and downstream success.

**2D Molecular Structure Parsing Poses a Significant Challenge.** Due to the inherently linear nature of SMILES notation, LLMs generally perform well on molecules with extended one-dimensional chains. However, their accuracy declines sharply when processing complex polycyclic systems with intricate 2D topologies.

## C.3 Case Study for Molecule Optimization

Molecular Optimization Tasks involve improving three physicochemical properties (QED, Solubility, LogP) and three protein-related activation capabilities (DRD2, JNK3, GSK3- $\beta$ ). Since large language models perform poorly in optimizing protein-related activations, we focus on their ability to optimize physicochemical properties. Table 3 presents the optimization results of three LLMs, including Gemini-2.5-pro, Qwen3-235B, and llama3.3-70B, revealing two key observations:

**LLMs exhibit significant potential in this task.** Despite the inherent difficulty of molecular optimization, LLMs exhibit significant potential in this task. We observed that these models introduce diverse functional groups, including halogens, aldehydes, hydroxyls, and amines, indicating broad chemical adaptability. However, some modifications led to negative optimization, likely due to limited understanding of the underlying physicochemical principles—a gap that could be addressed through targeted training.

**Commercial LLMs demonstrate bolder optimization strategies compared to open-source models.** For instance, Gemini-2.5-pro frequently performs skeleton-level modifications (e.g., additions or deletions), whereas Qwen3-235B and llama3.3 tend toward conservative insertions with minimal structural changes. This contrast highlights the greater flexibility and potential of commercial LLMs in molecular optimization.

Table 6: The case study for functional-group addition, deletion, and substitution in the molecule editing task. For better comparison, we visualize the predicted results from Gemini-2.5-pro (reasoning LLM), Qwen3-235B (non-reasoning LLM), and llama3.3-70B (non-reasoning LLM) and show the outstanding chemical reasoning ability of Gemini compared to other open-sourced LLMs.

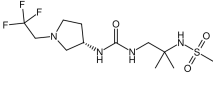
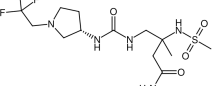
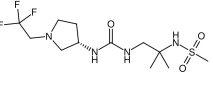
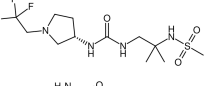
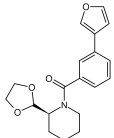
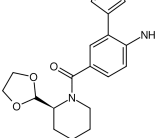
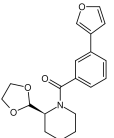
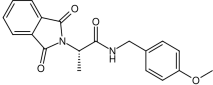
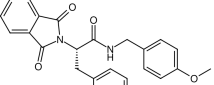
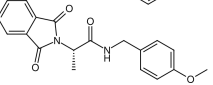
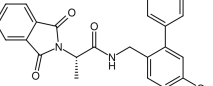
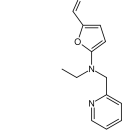
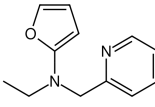
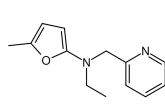
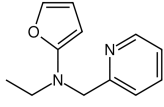
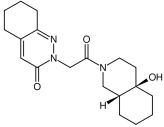
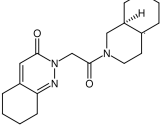
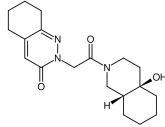
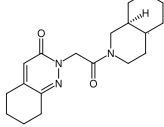
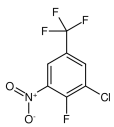
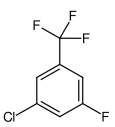
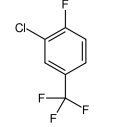
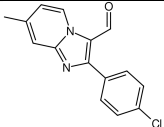
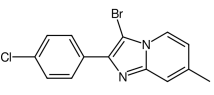
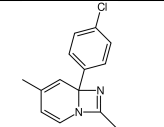
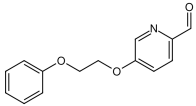
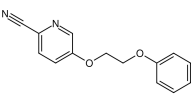
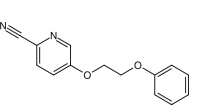
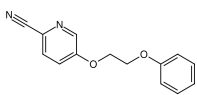
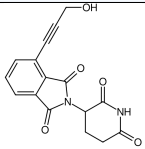
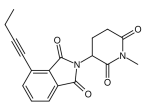
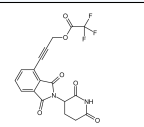
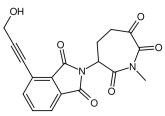
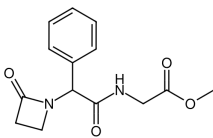
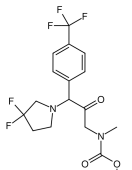
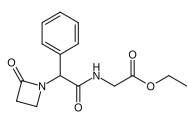
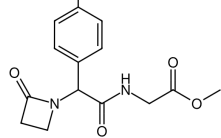
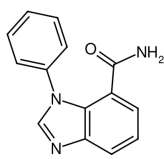
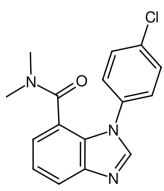
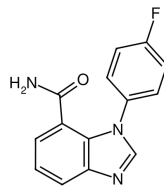
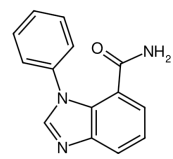
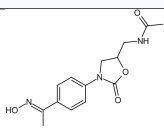
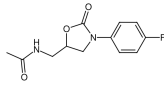
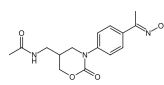
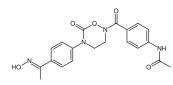
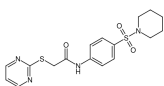
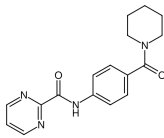
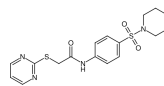
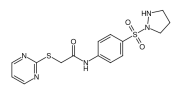
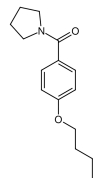
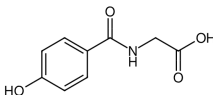
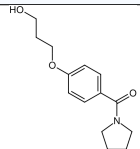
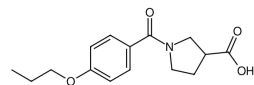
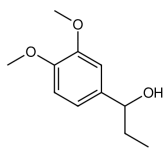
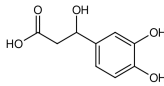
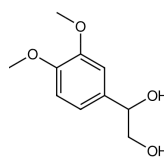
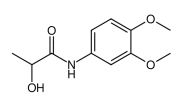
Instruction	Source Molecule	Gemini-2.5-pro	Qwen3-235B	Llama3.3-70B
<b>Add Functional Groups</b>				
Add the <b>amide group</b> while keeping the molecule scaffold unchanged.				
Add the <b>amine group</b> while keeping the molecule scaffold unchanged.				<b>Invalid SMILES</b>
Add the <b>benzene ring group</b> while keeping the molecule scaffold unchanged.				
<b>Delete Functional Groups</b>				
Delete <b>aldehyde group</b> while keeping the molecule scaffold unchanged.				
Delete <b>hydroxyl group</b> while keeping the molecule scaffold unchanged.				
Delete <b>nitro group</b> while keeping the molecule scaffold unchanged.			<b>Invalid SMILES</b>	
<b>Substitute Functional Groups</b>				
Remove <b>aldehyde group</b> and add <b>halo group</b> for the molecule.				<b>Invalid SMILES</b>
Remove <b>aldehyde group</b> and add <b>halo group</b> for the molecule.				

Table 7: The case study for Molecule Optimizations.

Source Molecule	Gemini-2.5-pro	QWen3-235B	Llama3.3-70B
<i>LogP Optimization</i>			
	 $\Delta = 1.16$	 $\Delta = 0.51$	 $\Delta = -3.76$
	 $\Delta = 1.68$	 $\Delta = 0.68$	 $\Delta = -0.39$
	 $\Delta = 0.68$	 $\Delta = 0.01$	 $\Delta = 0.0$
<i>QED Optimization</i>			
	 $\Delta = 0.38$	 $\Delta = 0.01$	 $\Delta = -0.03$
	 $\Delta = 0.34$	 $\Delta = 0.0$	 $\Delta = -0.03$
<i>Solubility Optimization</i>			
	 $\Delta = 3.47$	 $\Delta = 0.87$	 $\Delta = 0.48$
	 $\Delta = 1.08$	 $\Delta = 0.87$	 $\Delta = 0.52$

## 1008 D Task Example

1009 To better demonstrate the data structure of ChemCoTBench and the large-scale CoT dataset, we  
1010 conducted visualizations of representative samples from four distinct tasks: molecule understanding,  
1011 molecule editing, molecule optimization, and reaction prediction. As illustrated in Figure. 6, Figure. 7,  
1012 Figure. 8 and Figure. 10, each figure presents sample cases from different tasks, with text highlighted  
1013 in red indicating the chemical-specific prompt design.

**Question example for Molecule Understanding**

You are a chemical assistant. Please Determine whether the ring\_system\_scaffold is in the Molecule. Input: a molecule's SMILES string, a Ring System Scaffold. Output: yes / no.

**Definition: The ring system scaffold consists of one or more cyclic (ring-shaped) molecular structures**

Source Molecule: CC(C)n1cnc2c(NCc3ccc(-c4cccc4)cc3)nc(N(CCO)CCO)nc21, IUPAC of Source Molecule: 2-[2-hydroxyethyl]-[6-[(4-phenylphenyl)methylamino]-9-propan-2-yl]purin-2-yl]amino]ethanol. Ring system scaffold: c1ccc(-c2ccccc2)cc1.

Your response must be directly parsable JSON format:

```
{
  "input_structure": "original input structure",
  "molecule_structure_analysis": "describe the structure of the input Molecule",
  "scaffold_analysis": "describe the ring system scaffold",
  "matching_analysis": "matching the scaffold with the molecule",
  "output": "Yes / No"
}
```

DO NOT output other text except for the answer. If your response includes ``json``, regenerate it and output ONLY the pure JSON content.

Figure 6: Task example for molecule understanding subtask: Ring System Counting Task.

#### Question example for Molecule Editing

You are a chemical assistant. Given the SMILES structural formula of a molecule, help me add a specified functional group and output the improved SMILES sequence of the molecule. Input: Molecule SMILES string, Functional Group Name. Output: Modified Molecule SMILES string.

Source Molecule: O=S(=O)(Cc1nc(-c2cccs2)no1)c1ccc2ccccc2n1, Instruction: Modify the molecule by adding an aldehyde.

Your response must contain the step-by-step reasoning, and must be directly parsable JSON format:

```
{
  "molecule_analysis": "[your reasoning] Analyze the functional groups and other components within the molecule",
  "function_group_introduce_strategy": "[your reasoning] Determine how and at which site the new group can be most reasonably added",
  "feasibility_analysis": "[your reasoning] Assess the chemical viability of the proposed modification",
  "output": "Modified Molecule SMILES"
}
```

DO NOT output other text except for the answer. If your response includes ``json``, regenerate it and output ONLY the pure JSON content..

Figure 7: Task example for molecule editing subtask: Functional-Group Adding Task.

#### Question example for Molecule Optimization

You are a chemical assistant, Optimize the Source Molecule to improve the **GSK3-beta property (Glycogen Synthase Kinase 3-beta Inhibition)** while following a structured intermediate optimization process. IUPAC names are provided to resolve ambiguities in SMILES. For functional groups, IUPAC takes priority over SMILES. **Note these key group distinctions which are difficult to distinguish (1) Piperazine (1,4-diazacyclohexane): C1CNCCN1 (2) Piperidine (azinane): C1CCNCC1 (3) Pyrrole (azole): C1=CC=CN1**

Source Molecule: c1ccc(-c2cc(NCc3ccnc3)n3nccc3n2)cc1, IUPAC of Source Molecule: 5-phenyl-1-N-(pyridin-3-ylmethyl)pyrazolo[1,5-a]pyrimidin-7-amine.

Always output in strict, raw JSON format. Do NOT include any Markdown code block wrappers (e.g., ``json`` or ````). Your response must be directly passable JSON format:\n

```
{
  "Structural Analysis of Source Molecule": "",
  "Property Analysis": "",
  "Limitation in Source Molecule for Property": "",
  "Optimization for Source Molecule": "",
  "Final Target Molecule": "SMILES",
}
```

DO NOT output other text except for the answer. If your response includes ``json``, regenerate it and output ONLY the pure JSON content.

Figure 8: Task example for molecule optimization subtask: Optimizing GSK-3 $\beta$  Task.

### Question example for Next Elementary-step Product Prediction

We have one typical reaction (

**reaction class:** 'Bromo Sonogashira coupling',

**starting reactants:** 'CCOC(=O)C(OC(C)(C)C)c1c(C)cc2ccc(Br)cc2c1-c1ccc(Cl)cc1.C#CC(C)(C)O',

**reagents:** 'CCN(CC)CC.C1CCOC1.CCOC(=O)C(OC(C)(C)C)c1c(C)cc2ccc(Br)cc2c1-c1ccc(Cl)cc1.C#CC(C)(C)O.[Cl-].[Cu]I.[NH4+]',

**reaction condition:** 'Reaction with Pd coordinated with 3 or 4 ligands'

).

Here are the previous elementary reaction steps:

Elementary Step 1: {

"reactants":

c1ccc([PH])(c2ccccc2)(c2ccccc2)[Pd]([PH](c2ccccc2)(c2ccccc2)c2ccccc2)([PH](c2ccccc2)(c2ccccc2)c2ccccc2)[PH](c2ccccc2)(c2ccccc2)c2ccccc2)cc1,

"products":

c1ccc([PH])(c2ccccc2)(c2ccccc2)[Pd]([PH](c2ccccc2)(c2ccccc2)c2ccccc2)[PH](c2ccccc2)(c2ccccc2)c2ccccc2)cc1.c1ccc(P(c2ccccc2)c2ccccc2)cc1,

"step annotation": Ligand leaving,

}

Elementary Step 2: {

"reactants":

c1ccc([PH])(c2ccccc2)(c2ccccc2)[Pd]([PH](c2ccccc2)(c2ccccc2)c2ccccc2)[PH](c2ccccc2)(c2ccccc2)c2ccccc2)cc1,

"products":

c1ccc([PH])([Pd][PH](c2ccccc2)(c2ccccc2)c2ccccc2)(c2ccccc2)c2ccccc2)cc1.c1ccc(P(c2ccccc2)c2ccccc2)cc1,

"step annotation": Ligand leaving,

}

Now, we want to predict the next elementary reaction step.

Currently we know the basic information:

"current\_step\_info": {

"reactants": [Cu]I.C#CC(C)(C)O,

"step annotation": Copper activation,

}

Under the same reaction condition and reagents, please give me the products of the next step element reaction. Just return the SMILES of prediction.

Your response must contains directly parsable JSON format:

```
{
  "pred_smi": str
}
```

Figure 9: Task example for mechanism prediction subtask: Next Elementary-step Product Prediction.

### Question example for Mechanism Route Selection

For reaction class: 'Carboxylic acid + amine condensation',  
under the condition of 'Condensation using BOP' and given reagents (written in  
SMARTS format) '[#8]=[#6]-[#8].[#7,#16,#8].[#7]-[#8]-[P+]',  
which following description is the correct elementary reaction stages description,  
considering the mechanism of this type of reaction?

Choices:

**A:** Carboxylic acid deprotonation → Reaction of carboxylic acid and HATU/HBTU →  
Addition of HOBt (1-hydroxybenzotriazole) into carboxylic acid-HATU/HBTU →  
Amine attacks HOBt-carboxylic acid complex → Proton exchange between amide and  
HOBt

**B:** Proton exchange → Formation of a single bond between carboxylic acid and  
protonated DCC → Addition of amine (thiol) into carboxylic acid-DCC complex →  
Cleavage into amide and urea → Proton exchange between amide and urea

**C:** Carboxylic acid deprotonation → Reaction of carboxylic acid and CDI → Addition  
of imidazole into carboxylic acid-CDI → Amine attacks imidazole-carboxylic acid  
complex → Proton exchange between amide and imidazole

**D:** Addition of alcohol under the acidic conditions / deprotonation of alcohol →  
Neutralization of protonated ester / Addition of alcohol under the basic conditions

**E:** Proton exchange → Formation of a single bond between carboxylic acid and  
protonated DCC → Addition of HOBt (1-hydroxybenzotriazole) into carboxylic acid-  
DCC complex → Amine attacks HOBt-carboxylic acid complex → Proton exchange  
between amide and HOBt

**F:** Deprotonation of carboxylic acid → Nucleophilic substitution

**G:** Carboxylic acid deprotonation → Reaction of carboxylic acid and BOP → Addition  
of HOBt (1-hydroxybenzotriazole) into carboxylic acid-HATU/HBTU → Amine attacks  
HOBt-carboxylic acid complex → Proton exchange between amide and HOBt

**H:** Addition of amine into carboxylic acid → Deprotonation of amine → Hydroxide  
ion leaves

**I:** Addition to thionyl chloride → Addition of chloride → Pseudo-pericyclic expulsion  
of SO<sub>2</sub>, HCl → Nucleophilic addition → Nucleophilic addition → Deprotonation

**J:** Protonation of carbonyl or deprotonation of alcohol → Alcohol addition to carbonyl  
→ Protonation or deprotonation of complex → Water or hydroxide ion leaving →  
Proton exchange.

Return the choice (capital letter) in JSON format:

```
{  
  "choice": str # (e.g. 'A'/'B')  
}
```

Figure 10: Task example for mechanism prediction subtask: Mechanism Route Selection (MechSel).