KGRxn-LLM: Knowledge Graph Enhanced Large Language Models for Molecular Reaction Reasoning

Weichen Liu[1] **Qiyao Xue**[2] **Yuyang Wu**[3]
Olexandr Isayev[3] **Natasa Miskov-Zivanov**[1]
[1]University of Pittsburgh, [2]UNC-Chapel Hill, [3]Carnegie Mellon University
weichenliu@pitt.edu, nmzivanov@pitt.edu

Abstract

Large language models (LLMs) demonstrate strong general language capabilities but remain limited in chemical reasoning, particularly for tasks requiring structured, mechanistic understanding of molecular reactions. We present **K**nowledge **G**raph **R**eaction **LLM** (KGRxn-LLM), a framework that augments LLMs with a hierarchical chemical knowledge graph (KG) to ground reasoning in molecular transformations and reaction patterns. Existing benchmarks primarily emphasize reaction or molecular fact recall, providing limited assessment of reaction-level mechanistic reasoning. To address this gap, we introduce **KGRxn-Bench**, a benchmark of 1,200 questions designed to evaluate LLMs on reaction-centric reasoning tasks, including functional group identification, reaction type classification, and product and reagent prediction. Experimental results show that our approach of grounding LLMs in structured KG substantially improves performance across multiple tasks and model backbones, outperforming domain specific fine-tuned models on KG-covered splits and most hold-out splits.[1]

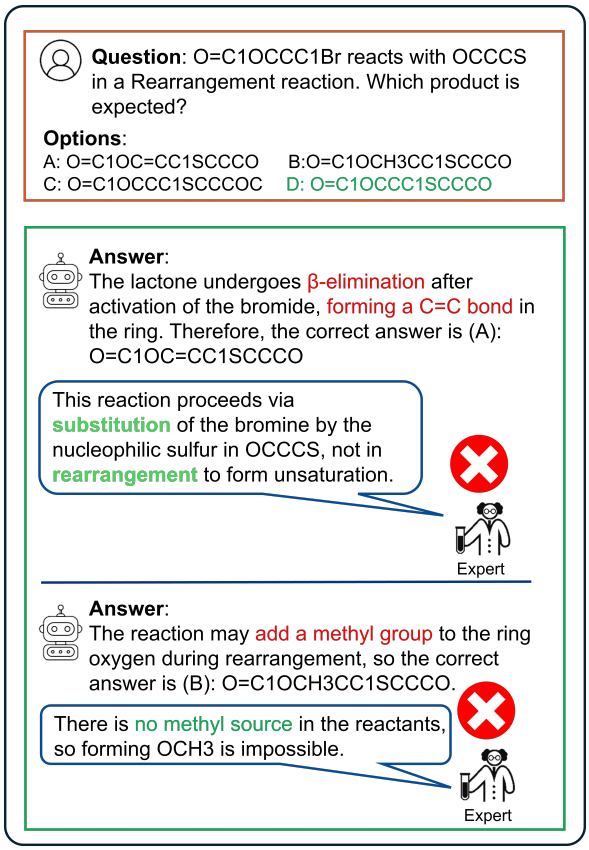

Figure 1: Illustration of hallucination problems of LLM doing molecular reaction reasoning.

1 Introduction

Recent advances in LLMs have led to remarkable progress on a broad spectrum of common-sense and general purpose language understanding tasks (Touvron et al., 2023; Achiam et al., 2023; Bai et al., 2023). Despite this success, LLMs continue to face significant challenges applied to chemistry, where domain knowledge is highly structured and governed by strict chemical rules rather than simple logic correlations (Bran et al., 2023; Jiménez-Luna et al., 2021). Prior efforts have extended LLMs to chemical domains such as molecular property question answering (QA), molecule caption generation, and molecular description tasks based on SMILES, which is string representation of molecular structures using predefined syntax (Lu et al., 2024; Laghuvarapu et al., 2025), demonstrating that LLMs can learn to generate syntactically valid chemical text by fine-tuning on large scale related corpus. However, recent evaluations show that these models remain fundamentally limited in chemical reaction reasoning, often misidentifying functional groups, confusing reaction types, and failing to track structural transformations across reactants and products, especially on tasks that require multi-step reasoning about reaction mechanisms or functional group changes (Runcie et al., 2025; Chen et al., 2025; Zhang et al., 2025).

[1]Code and benchmark are available at https://github.com/pitt-miskov-zivanov-lab/KGRxn-LLM.

Proceedings of the 25th Workshop on Biomedical Language Processing (BioNLP 2026), pages 268–281
July 3-4, 2026 ©2026 Association for Computational Linguistics

Current LLMs face hallucination problems in molecular reaction reasoning. Domain-specific models like ChemBERTa, MolT5, and Chemformer (Chithrananda et al., 2020; Edwards et al., 2022; Irwin et al., 2022) benefit from large-scale SMILES pretraining but these models may produce chemically invalid conclusions despite generating explanations that appear internally coherent. As shown in Figure 1, the first answer is confused about the "Rearrangement" reaction mechanism and the second answer involves wrong functional group source in reactant. A knowledge graph helps address this limitation by providing explicit chemical concepts as structured evidence for reasoning. Instead of relying only on pretrained knowledge, the model can retrieve grounded facts about molecules, functional groups, and reactions, which reduces hallucination and makes the reasoning process consistent with chemical rules.

Another fundamental challenge in this domain is the lack of benchmarks that explicitly evaluate LLMs' reasoning ability about molecular reaction transformations. Existing USPTO-based benchmarks (Chen et al., 2024; Yuan et al., 2025) focus primarily on forward reaction prediction or retrosynthesis, where success is measured by whether a model generates the correct product or SMILES representation. While these tasks are valuable, they do not assess whether a model understands the underlying transformation mechanisms or how functional groups evolve between reactants and products. On the other hand, chemical QA datasets such as MoleculeQA (Lu et al., 2024) and MolTextQA (Chen et al., 2025) focus on factual property comprehension, leaving reaction level reasoning ability such as identifying reaction types, tracking structural changes, or explaining mechanism relevant functional group operations largely untested.

To address these gaps, we introduce KGRxn-LLM, a knowledge graph(KG) integrated framework designed to advance knowledge-grounded molecular reasoning with LLMs. Our contribution can be summarized as follows:

- KGRxn-LLM introduces a hierarchical chemical knowledge graph with multiple layers, containing molecules, functional groups, reaction types and reaction instances. This KG encodes both chemical relationships and structural attributes of the chemical concept, forming an interpretable knowledge substrate.
- We propose KGRxn-Bench, a molecular reasoning benchmark built from curated USPTO reactions and annotated with functional group mappings and reactant-to-product transformations. The benchmark comprises tasks that evaluate reaction type identification, functional group transformation reasoning, and product prediction.
- We also propose an agent-based retrieval workflow that efficiently extracts transformation-relevant knowledge such as functional groups, common reaction patterns, similar molecules and reaction analogues, and uses this information to ground LLM reasoning, reducing hallucinations and improving accuracy.

2 Related work

LLMs for Biochemical Reasoning. The application of LLMs in biochemistry and computational biology has progressed rapidly, shifting from general text processing to specialized scientific discovery and molecular reasoning (Han et al., 2025; Usuyama et al., 2025). Recent paradigms treat biological molecules and small organic compounds as distinct text structures, allowing LLMs to excel in tasks ranging from protein structure prediction and *de novo* protein design to complex genomic analyses. For example, recent agent-based workflows have integrated LLMs to generate interpretable hypotheses for optimizing protein binding motifs and biochemical structures (Akke et al., 2025). Furthermore, the deployment of specialized multi-agent systems has enabled the autonomous execution of complex biochemical tasks, such as automating molecular dynamics simulations through tools (Campbell et al., 2025).

Parallel to these agentic frameworks, the emergence of advanced reinforcement learning-based reasoning models, such as DeepSeek-R1 and GPT-o series models (Guo et al., 2025; Jaech et al., 2024) has pushed the boundaries of zero-shot chemical intelligence, demonstrating robust capabilities in structural elucidation and the direct interpretation of molecular trends (Cui et al., 2025). While these models demonstrate impressive proficiency in broad biochemical property prediction and hypothesis generation, they continue to struggle with the rigorous, step-by-step mechanistic reasoning required to track atomic-level functional group transformations during chemical reactions. This fundamental limitation underscores the necessity for structurally knowledge-grounded approaches.

Knowledge Graph Guided LLM Reasoning. To mitigate hallucination and improve the factual reliability of LLMs, knowledge graph integration for Retrieval-Augmented Generation (RAG) has been employed (Lewis et al., 2020; Pan et al., 2024). Broadly, LLMs incorporate KGs to enhance reasoning through three primary mechanisms, including representation fusion, prompt augmentation, and tool-augmented querying (Pan et al., 2024). In representation fusion, KG network embeddings are injected directly into the LLMs' latent space during pre-training or fine-tuning, explicitly aligning textual tokens with structural entities to improve the model's internal knowledge representations (Yasunaga et al., 2022). Conversely, prompt-based retrieval methods, such as GraphRAG, dynamically extract multi-hop subgraphs, community summaries, or entity-relation triples from extensive knowledge bases, appending this structured context to the LLMs' input to guide explicit, step-by-step reasoning (Edge et al., 2024). To handle more complex logical queries, recent agentic frameworks like Think-on-Graph (ToG) and Reasoning-on-Graphs (RoG) empower LLMs to act as autonomous agents that iteratively explore graph structures by generating executable SQL queries to traverse multi-hop relational pathways before synthesizing an answer (Sun et al., 2023; Luo et al., 2023).

Within biochemical domains, these general frameworks have been successfully adapted by linking LLMs with specialized KGs such as UMLS or DrugBank to advance relational tasks like biomedical question answering and drug-target interaction prediction (Jin et al., 2024). Even though these domain-specific graphs excel at retrieving facts, they predominantly model knowledge as static nodes. Consequently, they remain unsuitable for capturing the dynamic, atomic-level transformations and mechanistic pathways required for complex chemical reaction reasoning. Our framework solves this problem with agentic retrieving algorithm, leveraging KG with comprehensive relationships of molecules and reactions.

Molecular Reasoning Benchmarks. The rapid integration of LLMs into biochemistry has motivated the development of specialized benchmarks for systematically evaluating their domain knowledge and reasoning capabilities. Recent benchmarks such as ChemLLMBench and ChemIQ (Guo et al., 2023; Runcie et al., 2025) assess a broad range of general chemistry tasks, including molecular property prediction, IUPAC to SMILES translation, and factual knowledge recall. To evaluate deeper structural understanding beyond conventional question answering, more recent benchmarks have begun to focus on fine grained and verifiable structural reasoning tasks. For example, FGBench introduces a rigorous evaluation framework for property reasoning at the functional group level, testing whether an LLM can interpret both single group and multi group structural interactions (Liu et al., 2025). ChemCoTBench tests stepwise molecular editing and optimization (Li et al., 2025), and MolErr2Fix measures the ability to detect and explain semantic hallucinations in molecular descriptions (Wu et al., 2025). However, these benchmarks provide limited coverage of reaction-centric reasoning, particularly mechanistic understanding of reactant-to-product transformations.

3 KGRxn-LLM Framework

3.1 Data Processing and Annotation

Preliminary Processing In retrieval-augmented settings, an excessively large graph may introduce noisy evidence, reduce retrieval precision, and degrade downstream reasoning quality (Gao et al., 2023). Therefore, we balance graph coverage with retrieval reliability. Following prior reaction prediction work (Coley et al., 2019), we extract 32,000 reactions from the USPTO reaction dataset (Marco et al., 2015). Among them, 30,000 reactions are used to construct the knowledge graph and the KG-covered portion of KGRxn-Bench, while the remaining 2,000 reactions are held out for constructing the benchmark hold-out split.

Starting from raw USPTO records, we first convert each reaction into a structured instance through JSON parsing and schema normalization, extracting reactants, products, and reaction-level metadata into a unified format. As illustrated in Figure 2, we then use RDKit to canonicalize molecular structures, trace atom correspondences between reactants and products, and identify functional groups in each molecule. Based on the atom-mapping results, we characterize functional-group-level transformations by determining which groups are retained, removed, or introduced during the reaction. The processed reaction, molecule, and functional-group annotations are stored in a structured database, which serves as the basis for knowledge graph construction and the KGRxn-LLM reasoning framework. More details are provided in Appendix A.1.

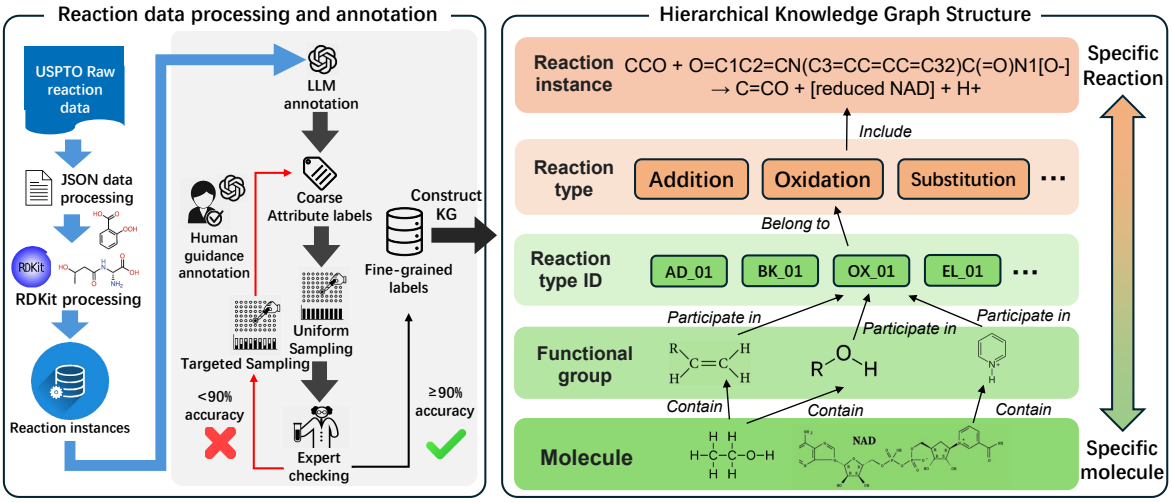

Figure 2: Overview of KGRxn-LLM data annotation pipeline and KG structure. Left: Raw USPTO reactions are processed into structured reaction instances, then annotated through an LLM-assisted pipeline with expert checking and resampling until reaching satisfactory accuracy. Right: Our hierarchical knowledge graph links molecules, functional groups, reaction type IDs, reaction types, and specific reaction instances across multiple levels.

Iterative Refinement Annotation Pipeline As shown in Figure 2, we employ an iterative human-in-the-loop pipeline to annotate KG entities and associated semantic labels in a scalable yet quality-controlled manner. In the first round, a strong closed-source LLM (ChatGPT 5) is used to produce coarse annotations for graph elements involved in knowledge graph construction, including reaction types, functional group categories and properties, and molecule-level properties. To assess annotation reliability, we uniformly sample a subset of instances for expert review and estimate annotation quality by empirical accuracy:

$$A = \frac{1}{|\mathcal{S}|} \sum_{x_i \in \mathcal{S}} \mathbf{1}[\hat{y}_i = y_i], \tag{1}$$

where $\mathcal{S}$ denotes the expert-reviewed subset, $\hat{y}_i$ is the LLM annotation, and y_i is the expert label.

When the verified accuracy falls below a predefined threshold of 90%, we perform a new round of targeted sampling that prioritizes instances from categories with higher observed error rates, while preserving the class composition of the previously reviewed subset to avoid severe sampling drift. Specifically, for each category c, we assign a sampling weight: $w_c \propto \pi_c(1 + \lambda e_c)$, where π_c is the category proportion in the previous review subset, e_c is the corresponding expert-verified error rate, and λ controls the emphasis on error-prone categories. This strategy increases the chance of selecting rare reaction types, molecules with multiple plausible functional groups, and structurally similar labels that are easy to confuse, while preserving the overall class composition.

At each audit round, previous observed errors are used to refine the annotation protocol with corrected examples, category-specific constraints, and explicit decision rules for ambiguous cases. The affected instances are then re-annotated under the revised protocol and audited again. This audit–refinement cycle is repeated until the target accuracy is reached. We also record category-wise error rates to identify systematic failure modes, allowing the KG annotation pipeline to scale beyond fully manual labeling while reducing bias from the pre-annotation model.

3.2 Reaction Knowledge Graph Construction

We construct the reaction knowledge graph as a hierarchical structure that organizes chemical knowledge from concrete molecular structures to abstract reaction semantics. Instead of treating each reaction as an isolated reactant–product pair, the graph decomposes reaction knowledge into multiple interconnected layers, including molecules, functional groups, fine-grained reaction patterns, coarse reaction categories, and specific reaction instances. This design improves retrieval efficiency by explicitly linking structural evidence to higher-level reaction concepts.

At the lowest level, molecule nodes represent the specific compounds involved in reactions. These

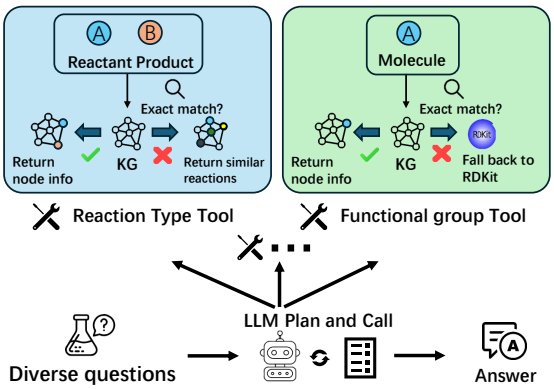

Figure 3: Task-specific tool calling and two-stage retrieval in KGRxn-LLM.

nodes preserve the original structural information and serve as the foundation of the graph. On top of this, functional group nodes capture chemically meaningful substructures, such as alcohols, alkenes, or carbonyl groups, and are linked to molecule nodes through "contain" relations. This layer provides a more reaction-relevant abstraction, since many reaction outcomes are determined by functional group transformations rather than whole-molecule identity.

Above the functional group layer, we introduce fine-grained Reaction Type ID nodes, which represent specific transformation patterns defined by characteristic local structural or functional-group changes. These nodes are further grouped into coarse reaction type nodes, such as oxidation, addition, and substitution, to provide a higher-level semantic taxonomy of reactions. Finally, reaction instance nodes represent individual reactions in the dataset and connect these abstract categories back to concrete examples. Overall, this hierarchical organization enables the graph to support both bottom-up reasoning from molecular structure to reaction type and top-down retrieval from reaction classes to plausible molecular transformations. More details can be found at Appendix A.2.

3.3 Knowledge Graph Retrieval Method

As illustrated in Figure 3, we formulate retrieval as an agentic tool-use process rather than a fixed lookup pipeline. Given a molecular reaction question, the LLM is prompted to determines which operation is required and then invokes the corresponding tool, such as functional group identification, reaction type inference, or other reaction-oriented utilities. Each tool is connected to the knowledge graph and follows a common two-stage retrieval strategy. It first attempts exact retrieval by matching canonicalized molecular or reaction representations to graph nodes, in which case the associated graph facts are returned directly as high-confidence evidence. If no exact match is available, the tool falls back to analogy-based retrieval at the functional-group level. Specifically, the query is represented by its consumed and formed functional group sets, and candidate reactions are ranked by the average Jaccard similarity (Travieso et al., 2024)

$$J_{\text{sim}} = \frac{1}{2}\left(\frac{|C_q \cap C_r|}{|C_q \cup C_r|} + \frac{|F_q \cap F_r|}{|F_q \cup F_r|}\right), \quad (2)$$

where C_q and F_q denote the consumed and formed functional groups of the query, and C_r and F_r denote those of a retrieved reaction. This retrieval criterion emphasizes analogous transformation patterns rather than superficial molecular similarity. For molecule-level queries such as functional group identification, the fallback is performed directly with RDKit-based functional group matching. The retrieved evidence is prepended to the original question and passed to the LLM for final answer generation in a single inference step. This design allows the model to adaptively choose the most relevant tool for each question while grounding its reasoning in explicit chemical evidence from either exact graph matches or structurally similar reaction analogues.

3.4 KGRxn-Bench Construction

We construct *KGRxn-Bench*, a benchmark designed to evaluate LLMs' ability on molecular reaction reasoning. As summarized in Table 1, the benchmark contains 1,200 questions derived from 986 unique reactions, covering 3,850 unique molecules, 10 reaction types, and 160 functional groups. The benchmark is designed to probe multiple levels of chemical reasoning, ranging from functional group recognition to reaction-level inference and condition prediction.

Task Definitions. KGRxn-Bench consists of four task types, each targeting a distinct aspect of reaction understanding.

Functional group identification requires the model to determine which functional group is present in a molecule represented by a SMILES string. This task evaluates whether the model can recognize chemically meaningful substructures from molecular representations.

Statistics	Number
Total Questions	1200
Unique Reactions	986
Unique Molecules	3850
Unique Reaction Types	10
Unique Functional Groups	160
Question Type Distribution	
Functional Group Identification (KG-covered)	272 (22.7%)
Functional Group Identification (hold-out)	100 (8.3%)
Reaction Type Classification (KG-covered)	243 (20.3%)
Reaction Type Classification (hold-out)	100 (8.3%)
Reaction Product Prediction (KG-covered)	151 (12.6%)
Reaction Product Prediction (hold-out)	100 (8.3%)
Reagent Prediction (KG-covered)	134 (11.2%)
Reagent Prediction (hold-out)	100 (8.3%)

Table 1: Key statistics of KGRxn-Bench.

Reaction type classification requires the model to infer the reaction category from reactant and product SMILES. Given the structural transformation, the model must assign the reaction to a predefined class such as substitution, oxidation, or addition. This task evaluates whether the model can abstract reaction semantics from explicit molecular changes.

Reaction product prediction requires the model to predict the expected product given two reactants and a specified reaction type. Unlike reaction classification, this task requires forward reasoning about how functional groups are transformed under a given reaction class.

Reagent prediction requires the model to infer plausible reagents (E.g.catalysts, solvents) from the observed reactant-to-product transformation. This task tests whether the model can associate structural changes with the reaction conditions typically required to realize them.

Question Answer Pair Generation. Questions are generated programmatically from the processed reaction data described in Section 3.1. For functional group identification, reaction type classification, and reagent prediction, multiple-choice distractors are generated by randomly sampling from the set of valid labels observed in the dataset. Specifically, we sample three incorrect options from the corresponding global label space, excluding the ground-truth answer, and combine them with the correct answer to form a multiple choice question with 4 options.

For reaction product prediction, we instead adopt a structure-aware mutation strategy to construct chemically plausible distractors. Rather than drawing unrelated molecules, we generate negative options by applying targeted modifications to the ground-truth product SMILES. These modifications include halogen substitution, replacement of carbonyl-adjacent groups, functional group swapping, aromatic position isomerization, double-bond toggling, substituent deletion, and small substituent addition. Mutation operators are applied iteratively until three unique and chemically valid distractors are obtained. This design makes the task substantially more challenging, as the incorrect options remain structurally similar to the correct answer and therefore require genuine mechanistic reasoning to distinguish.

KG-covered and Hold-out Splits. To evaluate both retrieval effectiveness and out-of-graph generalization, we partition the benchmark into *KG-covered* and *hold-out* subsets. The KG-covered subset contains questions derived from reactions included in the knowledge graph, allowing the retrieval module to access exact or closely related reaction evidence. In contrast, the hold-out subset is constructed from 2,000 reactions excluded during KG construction, ensuring that no exact reaction match is available in the graph.

Each task contains 100 hold-out questions, while the number of KG-covered questions varies across tasks according to the availability of eligible reactions. These two splits allow KGRxn-Bench to evaluate not only whether a model can benefit from graph-grounded retrieval, but also whether it can generalize to unseen reactions through higher level chemical similarity.

4 Experiments

4.1 Experiment Settings

Benchmarks We evaluate KGRxn-LLM on our proposed KGRxn-Bench together with two external benchmarks, FGBench (Liu et al., 2025) and SciKnowEval (Feng et al., 2024). KGRxn-Bench serves as our primary benchmark for molecular reaction reasoning, containing 1,200 multiple-choice questions spanning four tasks. We report results on both the KG-covered split and the hold-out split to evaluate the framework's ability to benefit from graph-grounded retrieval as well as to generalize beyond directly covered reactions. To further assess generalization beyond reaction-centered tasks, we additionally evaluate on FGBench and SciKnowEval, which probe broader biochemical understanding. Specifically, on FGBench we consider the Boolean tasks of single functional group impact,

	Functional Group		Reaction Product		Reaction Type		Reagent Prediction	
	KC	HO	KC	HO	KC	HO	KC	HO
# of samples	272	100	151	100	243	100	134	100
Vanilla Models								
Llama-3.1-8B	58.82	60.00	48.34	43.00	48.56	56.00	8.21	19.00
Qwen2.5-7B	63.97	67.00	48.34	63.00	46.09	64.00	14.18	13.00
nach0-base	19.12	20.00	18.54	12.00	6.17	6.00	9.70	7.00
LlaSMol-Mistral-7B	34.93	35.00	28.48	29.00	21.81	34.00	28.36	27.00
ChemLLM-7B	67.65	69.00	41.06	38.00	37.45	44.00	26.12	27.00
ChemLLM-7B-DPO	82.46	77.00	**73.51**	**71.00**	42.67	51.00	53.73	**40.00**
ChemDFM-v1.5-8B	67.65	77.00	43.71	46.00	61.32	<u>65.00</u>	25.37	33.00
KGRxn-LLM (Ours)								
Llama-3.1-8B + KG	<u>94.12</u>	**94.00**	<u>72.85</u>	56.00	**95.88**	64.00	**96.27**	37.00
Qwen2.5-7B + KG	**97.79**	<u>92.00</u>	**73.51**	<u>70.00</u>	<u>90.95</u>	**69.00**	<u>95.52</u>	<u>39.00</u>

Table 2: Performance in accuracy on KGRxn-Bench across four chemical reaction reasoning tasks. KC = KG-covered set, HO = Hold-out set. Best model results are in **bold**. Second best model results are <u>underlined</u>.

multiple functional group interactions and molecular comparisons. On SciKnowEval, we evaluate both multiple-choice and boolean questions in chemistry and biology. We use zero-shot accuracy as the evaluation metric for all tasks.

Baselines We compare KGRxn-LLM against both general-purpose and domain-specific language models. Specifically, we include instruction-tuned LLMs such as Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct as general model baselines, as well as chemistry-specialized models including nach0-base, LlaSMol-Mistral-7B, ChemLLM-7B, ChemLLM-7B-DPO and ChemDFM-v1.5-8B (Livne et al., 2024; Yu et al., 2024; Zhang et al., 2024; Zhao et al., 2025). All models are evaluated under a consistent zero-shot prompting setting without additional fine-tuning.

Implementation Details The knowledge graph is constructed from 30,000 USPTO reactions and hosted on a Neo4j Aura cloud instance. The graph contains overall 146,500 nodes and 269,981 relationships linking through the five-level graph hierarchy. For the retrieval module, the Jaccard similarity threshold is set to 0.3, with a maximum of 5 retrieved analogy reactions to ensure enough information guidance from KG. During generation, we use a temperature of 0.2, top-p of 0.9, and a maximum of 1024 new tokens for all question to accommodate long reasoning chains. All models are prompted with a zero-shot chain-of-thought format. More detailed prompt can be found in Appendix B.

4.2 Experimental results

As shown in Table 2, KGRxn-LLM consistently outperforms most baseline models across tasks on KGRxn-Bench. Performance gains are particularly substantial on the KG-covered split, where knowledge retrieval yields massive improvements over vanilla models. For instance, KG augmentation increases Llama-3.1-8B accuracy on functional group identification from 58.82% to 94.12%.This result confirms that when high-quality structured evidence is available, even general-purpose LLMs can leverage it effectively. Similar trends across other tasks confirm that structured knowledge effectively enhances mechanistic reasoning. On the hold-out split, KGRxn-LLM maintains consistent gains, demonstrating its capacity to generalize. Specifically, the augmented Qwen2.5-7B model improves functional group identification from 67.00% to 92.00%, suggesting that retrieved structural evidence provides a valuable reference even without exact graph matches.

Furthermore, Table 3 shows that KGRxn-LLM also improves performance on the external benchmarks FGBench and SciKnowEval. On FGBench, augmentation elevates functional group reasoning, notably raising the Qwen2.5-7B score from 39.33% to 49.64% on interaction queries. On SciKnowEval, the augmented Llama model improves from 88.24% to 95.76% on the biological boolean subset, reflecting broad gains across both chemistry and biology. Collectively, these findings indicate that the proposed framework benefits both specialized reaction

Model	FGBench			SciKnowEval			
	Single	Inter.	Comp.	Chem MCQ	Chem boolean	Bio MCQ	Bio boolean
Llama-3.1-8B	68.36	43.67	45.52	63.41	**78.67**	66.61	88.24
Qwen2.5-7B	57.34	39.33	**51.64**	68.28	66.67	76.23	83.36
Llama-3.1-8B + KG	**70.83**	43.95	46.94	71.71	77.25	68.35	**95.76**
Qwen2.5-7B +KG	68.50	**49.64**	48.32	**72.97**	76.53	**77.19**	92.79

Table 3: Performance in accuracy on external benchmarks: FGBench and SciKnowEval. In FGBench we consider the Boolean tasks of single functional group impact, multiple functional group interactions and molecular comparisons. In SciKnowEval we evaluate Multiple Choice Question(MCQ) and Boolean on Chemistry and biology subset. Best results per column are in **bold**.

reasoning and broader biochemical understanding through transferable structured knowledge.

4.3 Analysis

Overcoming Structural Hallucination A primary challenge for standard LLMs in chemistry is the tendency to generate structural hallucinations, where the model suggests products containing atoms or functional groups not present in the initial reactants. As illustrated in the comparative analysis, vanilla models often fail to maintain a strict atom inventory, erroneously adding substituents like methyl groups during rearrangement tasks despite no such source existing in the input SMILES. KGRxn-LLM mitigates this by providing pre-computed functional group annotations and structural transformations from KG. By grounding the reasoning process from different molecules to specific functional groups, the framework ensures that the model's internal logic remains consistent with chemical constraints.

Generalization Through Analogy The framework demonstrates a capacity of chemical reasoning by analogy, particularly evident in its performance on hold-out reaction sets where exact matches are unavailable in the KG. When a direct lookup fails, the agentic tool-use process falls back to a two-stage retrieval strategy that calculates Jaccard similarity based on consumed and formed functional group sets. This approach allows the LLM to identify analogous transformation patterns from similar reactions, such as the conversion of an alcohol to an ester, and apply those underlying rules to unseen molecular structures. Experimental results on the hold-out split confirm this effectiveness, as the retrieval of structurally similar reaction analogues provides a reliable reference that guides the model toward correct mechanistic pathways even in extrapolative settings.

Beyond Reaction Centered Tasks Result on FGBench reveals that vanilla models struggle with multi functional group interactions, while knowledge augmentation grounds these relationships in verified chemical rules. Similarly on SciKnowEval, the graph supplies foundational context that prevents factual error in broader chemistry and biology queries. Although it occasionally happens that in general knowledge evaluation settings performance drop compared to vanilla model when KG fail to retrieve any irrelevant context, this problem can be solved by include broader range of knowledge.

5 Conclusion

In this work, we present KGRxn-LLM, a knowledge graph augmented framework for molecular reaction reasoning that integrates hierarchical chemical knowledge with functional group level information retrieval. We also introduce KGRxn-Bench, a benchmark designed to evaluate reaction-centric reasoning beyond standard knowledge recalling tasks. Experimental results demonstrate that our framework of augmenting LLMs with structured chemical knowledge graph substantially improves performance across multiple reasoning tasks and model backbones, while also generalizing to broader biochemical understanding.

Limitations

This study has several limitations. First, the adopted taxonomy of ten coarse-grained reaction types provides only a simplified abstraction of organic chemistry. Second, the current knowledge graph, constructed from approximately 30K reactions, remains modest in scale relative to the vast space of known chemical reactions, which may constrain both coverage and retrieval effectiveness. Finally,

the framework has only been evaluated on mid-scale language models (7–8B parameters), and its scalability and effectiveness on larger models remain to be systematically investigated.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mattias Akke, Soojung Yang, Jurgis Ruza, Jinyeop Song, Elton Pan, and Rafael Gomez-Bombarelli. 2025. Bayesian optimization for biochemical discovery with llms.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*.

Quintina Campbell, Sam Cox, Jorge Medina, Brittany Watterson, and Andrew White. 2025. Mdcrow: Automating molecular dynamics workflows with large language models. *Machine Learning: Science and Technology*.

Shuan Chen, Ramil Babazade, Taewan Kim, Sunkyu Han, and Yousung Jung. 2024. A large-scale reaction dataset of mechanistic pathways of organic reactions. *Scientific Data*, 11(1):863.

Yufan Chen, Ching Ting Leung, Jianwei Sun, Yong Huang, Linyan Li, Hao Chen, and Hanyu Gao. 2025. Towards large-scale chemical reaction image parsing via a multimodal large language model. *arXiv preprint arXiv:2503.08156*.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*.

Connor W Coley, Wengong Jin, Luke Rogers, Timothy F Jamison, Tommi S Jaakkola, William H Green, Regina Barzilay, and Klavs F Jensen. 2019. A graph-convolutional neural network model for the prediction of chemical reactivity. *Chemical science*, 10(2):370–377.

Dong-Xu Cui, Shi-Yu Long, Yi-Xuan Tang, Yue Zhao, and Qiao Li. 2025. Can reasoning power significantly improve the knowledge of large language models for chemistry? based on conversations with llms. *Journal of Chemical Information and Modeling*, 65(18):9516–9527.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. *arXiv preprint arXiv:2204.11817*.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. 2024. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, and 1 others. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, and 1 others. 2023. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in neural information processing systems*, 36:59662–59688.

Yang Han, Ziping Wan, Lu Chen, Kai Yu, and Xin Chen. 2025. From generalist to specialist: A survey of large language models for chemistry. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1106–1123.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. 2022. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

José Jiménez-Luna, Francesca Grisoni, Nils Weskamp, and Gisbert Schneider. 2021. Artificial intelligence in drug discovery: recent advances and future perspectives. *Expert opinion on drug discovery*, 16(9):949–959.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2024. Genegpt: augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics*, 40(2):btae075.

Siddhartha Laghuvarapu, Namkyeong Lee, Chufan Gao, and Jimeng Sun. 2025. Moltextqa: A question-answering dataset and benchmark for evaluating multimodal architectures and llms on molecular structure–text understanding. *Journal of Data-centric Machine Learning Research*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Hao Li, He Cao, Bin Feng, Yanjun Shao, Xiangru Tang, Zhiyuan Yan, Li Yuan, Yonghong Tian, and Yu Li. 2025. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. *arXiv preprint arXiv:2505.21318*.

Xuan Liu, Siru Ouyang, Xianrui Zhong, Jiawei Han, and Huimin Zhao. 2025. Fgbench: A dataset and benchmark for molecular property reasoning at functional group-level in large language models. *arXiv preprint arXiv:2508.01055*.

Micha Livne, Zulfat Miftahutdinov, Elena Tutubalina, Maksim Kuznetsov, Daniil Polykovskiy, Annika Brundyn, Aastha Jhunjhunwala, Anthony Costa, Alex Aliper, Alán Aspuru-Guzik, and 1 others. 2024. nach0: multimodal natural and chemical languages foundation model. *Chemical Science*, 15(22):8380–8389.

Xingyu Lu, He Cao, Zijing Liu, Shengyuan Bai, Leqing Chen, Yuan Yao, Hai-Tao Zheng, and Yu Li. 2024. Moleculeqa: A dataset to evaluate factual accuracy in molecular comprehension. *arXiv preprint arXiv:2403.08192*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Alan C Marco, Amanda Myers, Stuart JH Graham, Paul D'Agostino, and Kirsten Apple. 2015. The uspto patent assignment dataset: Descriptions and analysis.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Nicholas T Runcie, Charlotte M Deane, and Fergus Imrie. 2025. Assessing the chemical intelligence of large language models. *Journal of Chemical Information and Modeling*, 66(1):216–227.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gonzalo Travieso, Alexandre Benatti, and Luciano da F Costa. 2024. An analytical approach to the jaccard similarity index. *arXiv preprint arXiv:2410.16436*.

Naoto Usuyama, Cliff Wong, Sheng Zhang, Tristan Naumann, and Hoifung Poon. 2025. Biomedical natural language processing in the era of large language models. *Annual Review of Biomedical Data Science*, 8.

Yuyang Wu, Jinhui Ye, Shuhao Zhang, Lu Dai, Yonatan Bisk, and Olexandr Isayev. 2025. Molerr2fix: Benchmarking llm trustworthiness in chemistry via modular error detection, localization, explanation, and correction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19365–19382.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. 2024. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*.

Shen Yuan, Shukai Gong, and Hongteng Xu. 2025. Uspto-llm: A large language model-assisted information-enriched chemical reaction dataset. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 817–820.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, and 1 others. 2024. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*.

Qiang Zhang, Keyan Ding, Tianwen Lv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, and 1 others. 2025. Scientific large language models: A survey on biological & chemical domains. *ACM Computing Surveys*, 57(6):1–38.

Zihan Zhao, Bo Chen, Ziping Wan, Lu Chen, Xuanze Lin, Shiyang Yu, Situo Zhang, Da Ma, Zichen Zhu, Danyang Zhang, and 1 others. 2025. Chemdfm-r: An chemical reasoner llm enhanced with atomized chemical knowledge. *arXiv e-prints*, pages arXiv–2507.

A Appendix

A.1 Preliminary Processing

Starting from raw USPTO reaction records in JSON format, we extract reaction identifiers, DOI provenance, atom-mapped reaction SMILES, and all input components (reactants, solvents, catalysts) and products into a unified tabular representation. All molecular SMILES are canonicalized using RDKit, and each molecule is assigned a deterministic identifier via the SHA-1 hash of its canonical SMILES (first 16 hex characters), ensuring consistent deduplication across reactions. We then apply 220 SMARTS-based substructure queries to detect functional groups in each molecule. When multiple patterns match the same atom set, a specificity ranking retains only the most descriptive label (e.g., *alcohol_secondary* over generic *alcohol*). For atom-mapped reactions, we identify the reaction center by comparing atom-level properties between reactant and product sides, then expand the changed-atom set by one bond radius. Only functional group instances overlapping this expanded center are regarded as reaction-participating groups. We then compute *formed* and *consumed* functional group sets by comparing (name, atom-map tuple) pairs across sides, which serve as input to a rule-based reaction type classifier containing 34 prioritized rules across seven priority levels that assign each reaction to one of ten coarse types. Reactions matching no rule are labeled *Unknown* and excluded. Finally, molecules originally annotated as reactants but who do not participate in the reaction center are reclassified as unchanged agents.

```
FG_SMARTS = {
    "alkene": Chem.MolFromSmarts("C=C"),
    "alkyne": Chem.MolFromSmarts("C#C"),
    "allene": Chem.MolFromSmarts("C=C=C"),
    "cumulene": Chem.MolFromSmarts("C(=C)=C"),
    "carbocation": Chem.MolFromSmarts("[C+]"),
    "carbanion": Chem.MolFromSmarts("[C-]"),
    "carbene": Chem.MolFromSmarts("[CX2]"),
    "borane": Chem.MolFromSmarts("[BH3]"),
    "boronic_acid": Chem.MolFromSmarts("B(O)O"),
    "boronic_ester": Chem.MolFromSmarts("B(O[#6])O[#6]"),
    "borinic_acid": Chem.MolFromSmarts("B(O)[#6]"),
    "borinic_ester": Chem.MolFromSmarts("B(O[#6])[#6]"),
    "borate_ester": Chem.MolFromSmarts("B(O[#6])(O[#6])O[#6]"),
    "alcohol": Chem.MolFromSmarts("[OX2H][#6]"),
    "alcohol_primary": Chem.MolFromSmarts("[CX4H2][OX2H]"),
    "alcohol_secondary": Chem.MolFromSmarts("[CX4H1]([#6])[OX2H]"),
    "alcohol_tertiary": Chem.MolFromSmarts("[CX4]([#6])([#6])[OX2H]"
      ),
    "alkoxide": Chem.MolFromSmarts("[O-][#6]"),
    "ether": Chem.MolFromSmarts("[OD2]([#6])[#6]"),
    "aryl_ether": Chem.MolFromSmarts("[OD2]([#6])[c]"),
    "phenol": Chem.MolFromSmarts("[c][OX2H]"),
    "phenolate": Chem.MolFromSmarts("[c][O-]"),
    "enol": Chem.MolFromSmarts("[CX3](=C)[OX2H]"),
    "enolate": Chem.MolFromSmarts("[CX3](=C)[O-]"),
    "enol_ether": Chem.MolFromSmarts("[CX3](=C)O[#6]"),
    "alkynol": Chem.MolFromSmarts("[CX2]#C[OX2H]"),
    "alkynolate": Chem.MolFromSmarts("[CX2]#C[O-]"),
    "alkynol_ether": Chem.MolFromSmarts("[CX2]#CO[#6]"),
    "ketone": Chem.MolFromSmarts("[CX3](=O)[#6]"),
    "aldehyde": Chem.MolFromSmarts("[CX3H1](=O)[#6]"),
    "ketene": Chem.MolFromSmarts("C=C=O"),
    "hemiketal": Chem.MolFromSmarts("[CX4]([OX2H])([OD2])[#6]"),
    "hemiacetal": Chem.MolFromSmarts("[CX4H1]([OX2H])([OD2])[#6]"),
    "ketal": Chem.MolFromSmarts("[CX4]([OD2])([OD2])[#6]"),
    "acetal": Chem.MolFromSmarts("[CX4H1]([OD2])([OD2])[#6]"),
    "carboxylic_acid": Chem.MolFromSmarts("[CX3](=O)[OX2H1]"),
    "carboxylate": Chem.MolFromSmarts("[CX3](=O)[O-]"),
    "ester": Chem.MolFromSmarts("[CX3](=O)O[#6]"),
    "organic_anhydride": Chem.MolFromSmarts("[CX3](=O)O[CX3](=O)"),
    "carboxylic_anhydride": Chem.MolFromSmarts("[CX3](=O)O[CX3](=O)"
      ),
    "organic_carbonate": Chem.MolFromSmarts("[CX3](=O)(O[#6])O[#6]"),

    "organic_hydroperoxide": Chem.MolFromSmarts("[OX2H]O[OX2]"),
    "organic_peroxide": Chem.MolFromSmarts("[OX2]O[OX2]"),
    "peroxyacid": Chem.MolFromSmarts("[CX3](=O)OO[OX2H]"),
    "methylenedioxy": Chem.MolFromSmarts("OCOc"),
    "ethylenedioxy": Chem.MolFromSmarts("OCCOc"),
    "epoxy": Chem.MolFromSmarts("C1OC1"),
    "silane": Chem.MolFromSmarts("[SiH4]"),
    "siloxane": Chem.MolFromSmarts("[Si]O[Si]"),
    "silyl_ether": Chem.MolFromSmarts("[Si]O[#6]"),
    "silyl_enol_ether": Chem.MolFromSmarts("[Si]O[C]=C"),
    "silyl_alkynol_ether": Chem.MolFromSmarts("[Si]O[C]#C"),
    "phosphine": Chem.MolFromSmarts("P([#6])([#6])[#6]"),
    "phosphonium": Chem.MolFromSmarts("[P+]"),
    "aminophosphine": Chem.MolFromSmarts("P(N)([#6])[#6]"),
    "phosphine_oxide": Chem.MolFromSmarts("P(=O)([#6])([#6])[#6]"),
    "phosphinic_acid": Chem.MolFromSmarts("P(=O)(O)[#6]"),
    "phosphinate": Chem.MolFromSmarts("P(=O)([O-])[#6]"),
    "phosphonic_acid": Chem.MolFromSmarts("P(=O)(O)(O)[#6]"),
    "phosphonate": Chem.MolFromSmarts("P(=O)(O[#6])([O-])[#6]"),
    "phosphite_ester": Chem.MolFromSmarts("P(O[#6])(O[#6])O[#6]"),
    "phosphinite": Chem.MolFromSmarts("P(O[#6])([#6])[#6]"),
    "phosphonite": Chem.MolFromSmarts("P(O[#6])(O[#6])[#6]"),
    "phosphodiester": Chem.MolFromSmarts("P(=O)(O[#6])O[#6]"),
    "phosphate_mono_ester": Chem.MolFromSmarts("P(=O)(O)(O)O[#6])"),
    "phosphate_tri_ester": Chem.MolFromSmarts("P(=O)(O[#6])(O[#6])O
      [#6]"),
    "phosphoramide": Chem.MolFromSmarts("P(=O)(N)(O[#6])O[#6]"),
    "thiophosphate": Chem.MolFromSmarts("P(=S)(O[#6])(O[#6])O[#6]"),
    "phosphonium_ylide": Chem.MolFromSmarts("[P+][C-]"),
    "fluoro": Chem.MolFromSmarts("[F]"),
    "chloro": Chem.MolFromSmarts("[Cl]"),
    "bromo": Chem.MolFromSmarts("[Br]"),
    "iodo": Chem.MolFromSmarts("[I]"),
    "alkyl_halide": Chem.MolFromSmarts("[CX4][F,Cl,Br,I]"),
    "vinyl_halide": Chem.MolFromSmarts("[CX3]=[CX3][F,Cl,Br,I]"),
    "aryl_halide": Chem.MolFromSmarts("[c][F,Cl,Br,I]"),
    "halamine": Chem.MolFromSmarts("N[Cl,Br,I,F]"),
    "sulfenyl_halide": Chem.MolFromSmarts("S[Cl,Br,I,F]"),
    "sulfinyl_halide": Chem.MolFromSmarts("S(=O)[Cl,Br,I,F]"),
    "sulfonyl_halide": Chem.MolFromSmarts("S(=O)(=O)[Cl,Br,I,F]"),
    "halosulfate": Chem.MolFromSmarts("OS(=O)(=O)[Cl,Br,I,F]"),
    "phosphoryl_halide": Chem.MolFromSmarts("P(=O)([Cl,Br,I,F])[Cl,
      Br,I,F]"),
    "phosphorus_halide": Chem.MolFromSmarts("P([Cl,Br,I,F])([Cl,Br,I,
      F])[Cl,Br,I,F]"),
    "acyl_halide": Chem.MolFromSmarts("[CX3](=O)[Cl,Br,I,F]"),
    "imidoyl_halide": Chem.MolFromSmarts("[CX3](=[NX2])[Cl,Br,I,F]"),

    "thioacyl_halide": Chem.MolFromSmarts("[CX3](=S)[Cl,Br,I,F]"),
    "organolithium": Chem.MolFromSmarts("[#6][Li]"),
    "organomagnesium": Chem.MolFromSmarts("[#6][Mg]"),
    "organoaluminium": Chem.MolFromSmarts("[#6][Al]"),
    "organozinc": Chem.MolFromSmarts("[#6][Zn]"),
    "organomercury": Chem.MolFromSmarts("[#6][Hg]"),
    "pyrrolic_N": Chem.MolFromSmarts("[nH]"),
    "pyridinic_N": Chem.MolFromSmarts("n"),
    "aromatic_O": Chem.MolFromSmarts("o"),
    "aromatic_S": Chem.MolFromSmarts("s"),
    "primary_amine": Chem.MolFromSmarts("[NX3H2][#6]"),
    "secondary_amine": Chem.MolFromSmarts("[NX3H1]([#6])[#6]"),
    "tertiary_amine": Chem.MolFromSmarts("[NX3]([#6])([#6])[#6]"),
    "ammonium_cation": Chem.MolFromSmarts("[NX4+]"),
    "amine_oxide": Chem.MolFromSmarts("[NX3+][O-]"),
    "enamine": Chem.MolFromSmarts("[NX3][CX3]=[CX3]"),
    "hydroxylamine": Chem.MolFromSmarts("NO"),
    "hemiaminal": Chem.MolFromSmarts("[CX4]([OX2H])([NX3])[#6]"),
    "hemiaminal_ether": Chem.MolFromSmarts("[CX4]([OD2])([NX3])[#6]"
      ),
    "thioaminal": Chem.MolFromSmarts("[CX4]([SX2H])([NX3])[#6]"),
    "thioaminal_ether": Chem.MolFromSmarts("[CX4]([SX2])([NX3])[#6]"
      ),
    "aminal": Chem.MolFromSmarts("[CX4]([NX3])([NX3])[#6]"),
    "primary_ketimine": Chem.MolFromSmarts("[CX3](=[NX2H])[#6]"),
    "secondary_ketimine": Chem.MolFromSmarts("[CX3](=[NX2][#6])[#6]"
      ),
    "primary_aldimine": Chem.MolFromSmarts("[CX3H1](=[NX2H])[#6]"),
    "secondary_aldimine": Chem.MolFromSmarts("[CX3H1](=[NX2][#6])
      [#6]"),
    "amidine": Chem.MolFromSmarts("N=C(N)[#6]"),
    "guanidine": Chem.MolFromSmarts("N=C(N)N"),
    "ketoxime": Chem.MolFromSmarts("[CX3](=N[OX2H])[#6]"),
    "aldoxime": Chem.MolFromSmarts("[CX3H1](=N[OX2H])[#6]"),
    "hydrazone": Chem.MolFromSmarts("[NX3][NX2]=[CX3]"),
    "amide": Chem.MolFromSmarts("[NX3][CX3](=O)[#6]"),
    "amidate_anion": Chem.MolFromSmarts("[NX3][CX3](=O)[O-]"),
    "imide": Chem.MolFromSmarts("[CX3](=O)N[CX3](=O)"),
    "carbamic_acid": Chem.MolFromSmarts("NC(=O)O"),
    "carbamate_ester": Chem.MolFromSmarts("NC(=O)O[#6]"),
    "carbamate_anion": Chem.MolFromSmarts("[NX3][CX3](=O)[O-]"),
```

```
    "azide": Chem.MolFromSmarts("N=[N+]=[N-]"),
    "azo": Chem.MolFromSmarts("N=N"),
    "hydrazine": Chem.MolFromSmarts("NN"),
    "acylhydrazine": Chem.MolFromSmarts("[CX3](=O)NN"),
    "cyanate": Chem.MolFromSmarts("OC#N"),
    "isocyanate": Chem.MolFromSmarts("N=C=O"),
    "nitrile": Chem.MolFromSmarts("C#N"),
    "isonitrile": Chem.MolFromSmarts("N#C"),
    "cyanamide": Chem.MolFromSmarts("N=C=N"),
    "carbodiimide": Chem.MolFromSmarts("N=C=N"),
    "nitrate_ester": Chem.MolFromSmarts("O[N+](=O)[O-]"),
    "nitrite_ester": Chem.MolFromSmarts("ON=O"),
    "nitro": Chem.MolFromSmarts("[NX3](=O)=O"),
    "nitroso": Chem.MolFromSmarts("N=O"),
    "nitrosamine": Chem.MolFromSmarts("N(N)=O"),
    "iminium_cation": Chem.MolFromSmarts("[CX3]=[NX3+]"),
    "nitrone": Chem.MolFromSmarts("[NX3+](=O)[O-]"),
    "nitronic_acid": Chem.MolFromSmarts("[CX3](=N(=O)O)O"),
    "urea": Chem.MolFromSmarts("NC(=O)N"),
    "azoxy": Chem.MolFromSmarts("N(=O)N"),
    "N_oxoammonium": Chem.MolFromSmarts("[NX4+]=O"),
    "hydroxamic_acid": Chem.MolFromSmarts("NC(=O)NO"),
    "hydroxamate": Chem.MolFromSmarts("NC(=O)N[O-]"),
    "azanide": Chem.MolFromSmarts("[N-]"),
    "nitronate": Chem.MolFromSmarts("[O-][N+](=O)[#6]"),
    "mercaptan": Chem.MolFromSmarts("[SX2H]"),
    "thiolate_anion": Chem.MolFromSmarts("[S-][#6]"),
    "organic_sulfide": Chem.MolFromSmarts("[SX2]([#6])[#6]"),
    "thioenol": Chem.MolFromSmarts("C=C[SX2H]"),
    "enedithiol": Chem.MolFromSmarts("C=C([SX2H])[SX2H]"),
    "thioenolate": Chem.MolFromSmarts("C=C[S-]"),
    "thioenol_ether": Chem.MolFromSmarts("C=C[SX2][#6]"),
    "organic_disulfide": Chem.MolFromSmarts("[#16X2]-[#16X2]"),
    "sulfenic_acid": Chem.MolFromSmarts("[#16X2][OX2H]"),
    "sulfenic_ester": Chem.MolFromSmarts("[#16X2][OX2][#6]"),
    "sulfenamide": Chem.MolFromSmarts("[#16X2][NX3]"),
    "sulfoxide": Chem.MolFromSmarts("[#16X3](=O)"),
    "sulfone": Chem.MolFromSmarts("[#16X4](=O)(=O)"),
    "sulfinylamine": Chem.MolFromSmarts("S(=O)N"),
    "sulfinic_acid": Chem.MolFromSmarts("S(=O)[OX2H]"),
    "sulfonic_acid": Chem.MolFromSmarts("S(=O)(=O)[OX2H]"),
    "sulfinate_ester": Chem.MolFromSmarts("S(=O)(O[#6])[#6]"),
    "sulfonate_ester": Chem.MolFromSmarts("S(=O)(=O)O[#6]"),
    "sulfinate_anion": Chem.MolFromSmarts("S(=O)([O-])[#6]"),
    "sulfonate_anion": Chem.MolFromSmarts("S(=O)(=O)[O-]"),
    "thiosulfonate": Chem.MolFromSmarts("S(=O)(=O)SS"),
    "organosulfite": Chem.MolFromSmarts("S(=O)(O[#6])O[#6]"),
    "organosulfate": Chem.MolFromSmarts("S(=O)(=O)(O[#6])O[#6]"),
    "dialkylsulfates": Chem.MolFromSmarts("OS(=O)(=O)O[#6]"),
    "sulfinamide": Chem.MolFromSmarts("S(=O)N[#6]"),
    "sulfonamide": Chem.MolFromSmarts("S(=O)(=O)N"),
    "sulfamic_acid": Chem.MolFromSmarts("NS(=O)(=O)O"),
    "sulfamate": Chem.MolFromSmarts("NS(=O)(=O)O[#6]"),
    "sulfamide": Chem.MolFromSmarts("NS(=O)(=O)N"),
    "thiocyanate": Chem.MolFromSmarts("S-C#N"),
    "isothiocyanate": Chem.MolFromSmarts("N=C=S"),
    "thioketone": Chem.MolFromSmarts("C(=S)[#6]"),
    "thioketene": Chem.MolFromSmarts("C=C=S"),
    "thial": Chem.MolFromSmarts("[CX3H1](=S)[#6]"),
    "thioamide": Chem.MolFromSmarts("NC(=S)[#6]"),
    "thiourea": Chem.MolFromSmarts("NC(=S)N"),
    "hemithioketal": Chem.MolFromSmarts("[CX4]([SX2H])([SX2])[#6]"),
    "hemithioacetal": Chem.MolFromSmarts("[CX4H1]([SX2H])([SX2])[#6]
      "),
    "dithiohemiketal": Chem.MolFromSmarts("[CX4]([SX2H])([SX2H])[#6]
      "),
    "dithiohemiacetal": Chem.MolFromSmarts("[CX4H1]([SX2H])([SX2H])
      [#6]"),
    "monothioketal": Chem.MolFromSmarts("[CX4]([SX2])([OD2])[#6]"),
    "monothioacetal": Chem.MolFromSmarts("[CX4H1]([SX2])([OD2])[#6]"
      ),
    "dithioketal": Chem.MolFromSmarts("[CX4]([SX2])([SX2])[#6]"),
    "dithioacetal": Chem.MolFromSmarts("[CX4H1]([SX2])([SX2])[#6]"),
    "carbothioic_S_acid": Chem.MolFromSmarts("C(=S)S"),
    "carbothioic_O_acid": Chem.MolFromSmarts("C(=S)O"),
    "thiol_form_thiocarboxylate": Chem.MolFromSmarts("C(=O)[S-]"),
    "thione_form_thiocarboxylate": Chem.MolFromSmarts("C(=S)[O-]"),
    "thiolester": Chem.MolFromSmarts("C(=O)S[#6]"),
    "thionoester": Chem.MolFromSmarts("C(=S)O[#6]"),
    "carbodithioic_acid": Chem.MolFromSmarts("C(=S)SS"),
    "carbodithioic_anion": Chem.MolFromSmarts("C(=S)S[S-]"),
    "carbodithioic_ester": Chem.MolFromSmarts("C(=S)S[#6]"),
    "monothiocarbonate": Chem.MolFromSmarts("C(=O)(O)S"),
    "xanthic_acid": Chem.MolFromSmarts("C(=S)(O)S"),
    "xanthate": Chem.MolFromSmarts("C(=S)(O[#6])S[#6]"),
    "xanthate_anion": Chem.MolFromSmarts("C(=S)(O[#6])[S-]"),
    "dithiocarbonate": Chem.MolFromSmarts("C(=S)(S)S"),
    "O_thiocarbamic_acid": Chem.MolFromSmarts("NC(=S)O"),
    "S_thiocarbamic_acid": Chem.MolFromSmarts("NC(=O)S"),
    "O_thiocarbamate": Chem.MolFromSmarts("NC(=S)O[#6]"),
    "S_thiocarbamate": Chem.MolFromSmarts("NC(=O)S[#6]"),
    "O_thiocarbamate_anion": Chem.MolFromSmarts("NC(=S)[O-]"),
    "S_thiocarbamate_anion": Chem.MolFromSmarts("NC(=O)[S-]"),
    "thioimidic_acid": Chem.MolFromSmarts("C(=S)N"),
    "thioimidate_anion": Chem.MolFromSmarts("C(=S)[N-]"),
    "thioimidate": Chem.MolFromSmarts("C(=S)N[#6]"),
    "dithiocarbamic_acid": Chem.MolFromSmarts("NC(=S)S"),
    "dithiocarbamate": Chem.MolFromSmarts("NC(=S)S[#6]"),
    "dithiocarbamate_anion": Chem.MolFromSmarts("NC(=S)[S-]"),
    "sulfonium_ion": Chem.MolFromSmarts("[S+]"),
    "sulfonium_ylide": Chem.MolFromSmarts("[S+][C-]"),
    "sulfoxonium_ylide": Chem.MolFromSmarts("[S+](=O)[C-]"),
}
```

Listing 1: Full FG_SMARTS dictionary used for functional group matching.

A.2 Knowledge Graph Construction

The knowledge graph is implemented in Neo4j and organized into five node types connected by six edge types, forming the hierarchical structure illustrated in Figure 2. At the base layer, **Molecule** nodes store each compound's canonical SMILES and a hash-based identifier, while **Solvent** nodes (a separate label) represent molecules identified as non-reactive participants. Above this, **FunctionalGroup** nodes represent the 220 named substructure patterns, linked to molecules via HAS_FUNCTION_GROUP edges. At the reaction abstraction layer, **ReactionType** nodes encode ten coarse categories (e.g., Oxidation, Substitution), and **ReactionTypeId** nodes capture the 34 fine-grained transformation rules, connected by BELONGS_TO edges. Functional groups are linked to their associated reaction type IDs through PARTICIPATES_IN edges annotated with the specific functional group transformation pattern (e.g., "alcohol + carboxylic_acid $\rightarrow$ ester"), while solvents are linked via SOLVENT_IN edges. At the top layer, **Reaction** nodes represent individual reaction instances, each storing the atom-mapped reaction SMILES, reactant and product molecule identifiers, formed and consumed functional group lists, and a textual mechanistic explanation. Reaction nodes are connected to their corresponding ReactionType via CLASSIFIED_AS edges, and to participant Molecule and Solvent nodes via PARTICIPATES_IN and SOLVENT_IN edges respectively. All nodes and edges are created using idempotent `MERGE` statements, ensuring that the graph can be incrementally updated without duplication.

B Zero-shot Prompt in Evaluation

All the system prompt for model to do zero-shot reasoning is shown below:

VANILLA PROMPTS (without Knowledge Graph)

— Functional Group Identification —

[System Prompt] You are an expert organic chemist. Your task is to identify which functional group is contained in the given molecule (provided as SMILES notation).

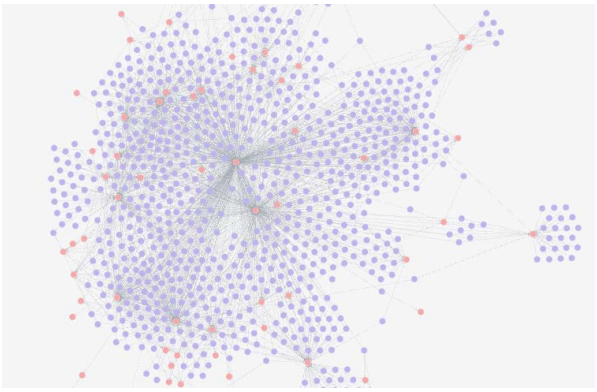

Figure 4: A sub-graph of 1000 relationships in the kG.

Think step by step: 1. First, analyze the SMILES string and identify the molecular structure 2. Identify the key functional groups present in the molecule 3. Compare with the given options 4. Select the correct answer

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

— Reaction Type Classification —

[System Prompt] You are an expert organic chemist. Your task is to identify the reaction type of a given chemical reaction (provided as SMILES notation).

Think step by step: 1. First, analyze the reactant and product SMILES strings and identify the molecular structures 2. Identify the key functional group transformations (which groups are consumed and which are formed) 3. Based on the transformations, determine the reaction type 4. Compare with the given options and select the correct answer

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

— Reaction Product Prediction —

[System Prompt] You are an expert organic chemist. Your task is to predict the product of a chemical reaction given the reactants and reaction type.

Think step by step: 1. First, analyze the reactant SMILES strings and identify the molecular structures 2. Consider the specified reaction type and what transformations it typically involves 3. Predict the product structure based on the reactants and reaction type 4. Compare with the given options and select the most likely product

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

— Reagent Prediction —

[System Prompt] You are an expert organic chemist. Your task is to predict the reagents (catalysts, solvents, or ancillary substances) needed for a given chemical reaction (provided as SMILES notation).

Think step by step: 1. First, analyze the reactant and product SMILES strings and identify the molecular structures 2. Identify the key functional group transformations (which groups are consumed and which are formed) 3. Based on the transformations, reason about what reagents, catalysts, or solvents would be needed 4. Compare with the given options and select the correct answer

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

KG-AUGMENTED PROMPTS (with Knowledge Graph)

— Functional Group Identification + KG —

[System Prompt] You are an expert organic chemist. Your task is to identify which functional group is contained in the given molecule.

You will be provided with: 1. The molecule in SMILES notation 2. Information retrieved from a Chemistry Knowledge Graph about the functional groups present in this molecule

Use the Knowledge Graph information to help you determine the correct answer. Think step by step: 1. Review the functional groups identified by the Knowledge Graph 2. Compare these with the given options 3. Select the correct answer based on the evidence

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

[Knowledge Graph Information] Molecule SMILES: smiles Source: source Functional groups found: comma-separated list

Please analyze the molecule and select the correct functional group from the options above.

— Reaction Type Classification + KG —

[System Prompt] You are an expert organic chemist. Your task is to identify the reaction type of a given chemical reaction (provided as SMILES notation).

You will be provided with: 1. The reaction in SMILES notation 2. Information retrieved from a Chemistry Knowledge Graph about functional group transformations or exact matches

Use the Knowledge Graph information to help you determine the correct reaction type. Think step by step: 1. Review the information from the Knowledge Graph (exact match or similar reactions) 2. Analyze the reactant and product structures and their functional group changes 3. Based on the transformations, determine the reaction type 4. Compare with the given options and select the correct answer

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

rag_context from get_reaction_type_rag()

Please analyze the reaction and select the correct reaction type from the options above.

— Reaction Product Prediction + KG —

[System Prompt] You are an expert organic chemist. Your task is to predict the product of a chemical reaction given the reactants and reaction type.

You will be provided with: 1. The reactants in SMILES notation and the reaction type 2. Information retrieved from a Chemistry Knowledge Graph about similar reactions or exact matches

Use the Knowledge Graph information to help you determine the correct product. Think step by step: 1. Review the information from the Knowledge Graph (exact match or similar reactions) 2. Analyze the reactant structures and reaction type 3. Consider the functional group transformations involved 4. Compare with the given options and select the most likely product

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

rag_context from get_reaction_product_rag()

Please analyze the reaction and select the correct product from the options above.

— Reagent Prediction + KG —

[System Prompt] You are an expert organic chemist. Your task is to predict the reagents (catalysts, solvents, or ancillary substances) needed for a given chemical reaction (provided as SMILES notation).

You will be provided with: 1. The reaction (reactants and product) in SMILES notation 2. Information retrieved from a Chemistry Knowledge Graph about exact matches or similar reactions with their reagents

Use the Knowledge Graph information to help you determine the correct reagent. Think step by step: 1. Review the information from the Knowledge Graph (exact match or similar reactions) 2. Analyze the reactant and product structures and the functional group changes 3. Based on the transformation, reason about what reagents would be needed 4. Compare with the given options and select the correct answer

You MUST end your response with your final answer in this exact format: **Answer: X** where X is A, B, C, or D.

[User Message] question

Options: options

rag_context from predict_reagent()

Please analyze the reaction and select the correct reagent from the options above.