

Structured Chemistry Reasoning with Large Language Models

Siru Ouyang¹ Zhuosheng Zhang² Bing Yan³ Xuan Liu¹ Yejin Choi^{4,5} Jiawei Han¹ Lianhui Qin^{5,6}

Abstract

Large Language Models (LLMs) excel in diverse areas, yet struggle with complex scientific reasoning, especially in the field of chemistry. Different from the simple chemistry tasks (e.g., molecule classification) addressed in previous studies, complex chemistry problems require not only vast knowledge and precise calculation, but also compositional reasoning about rich dynamic interactions of different concepts (e.g., temperature changes). Our study shows that even advanced LLMs, like GPT-4, can fail easily in different ways. Interestingly, the errors often stem not from a lack of domain knowledge within the LLMs, but rather from the absence of an effective reasoning *structure* that guides the LLMs to elicit the right knowledge, incorporate the knowledge in step-by-step reasoning, and iteratively refine results for further improved quality. On this basis, we introduce STRUCTCHEM, a simple yet effective prompting strategy that offers the desired guidance and substantially boosts the LLMs’ chemical reasoning capability. Testing across four chemistry areas—quantum chemistry, mechanics, physical chemistry, and kinetics—STRUCTCHEM substantially enhances GPT-4’s performance, with up to 30% peak improvement. Our analysis also underscores the unique difficulties of precise grounded reasoning in science with LLMs, highlighting a need for more research in this area. Code is available at <https://github.com/ozyyshr/StructChem>.

1. Introduction

Artificial intelligence (AI) holds the promise of transforming the field of chemistry (Baum et al., 2021), impacting

^{*}Equal contribution ¹University of Illinois Urbana-Champaign ²Shanghai Jiao Tong University ³New York University ⁴University of Washington ⁵Allen Institute for AI ⁶University of California San Diego. Correspondence to: Lianhui Qin <lianhui@ucsd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

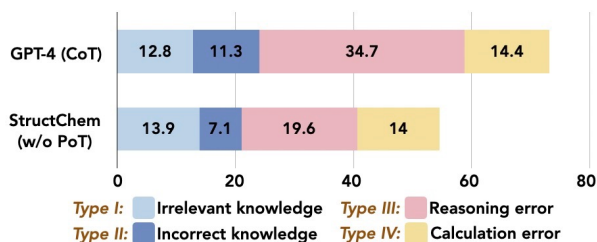


Figure 1. Proportions (%) of four error types (#errors / #all-cases) for GPT-4 and STRUCTCHEM. STRUCTCHEM substantially reduces the reasoning error.

various sectors including industrial production (Öztürk et al., 2020), pharmaceuticals (Singhal et al., 2023), and education (Graulich et al., 2022). Recent studies have shown promising results of large language models (LLMs) solving simple chemistry problems (Figure 2(a)), such as molecule classification (Edwards et al., 2022) and property prediction (Yang et al., 2019; Feinberg et al., 2018).

On the other hand, however, more complex chemistry reasoning problems still pose significant challenges to frontier LLMs like GPT-4. As shown in Figure 2(b), a complex problem requires not only understanding individual concepts (e.g., molecule property) as in previous tasks, but rather their rich dynamic interactions in different contexts, involving extensive domain knowledge (e.g., chemical formulae¹), precise scientific computing, and compositional step-by-step reasoning. As a result, LLMs are prone to different forms of errors when solving these problems, such as applying incorrect knowledge, making miscalculations, or following flawed reasoning processes, as illustrated in Figure 2(c).

Interestingly, as shown in Figure 1 and discussed in Section 5, LLMs oftentimes have encoded necessary knowledge for a given chemistry problem. The key shortfall, however, lies in the absence of a sophisticated reasoning structure that helps elicit the relevant knowledge from the LLMs, and guides them to perform precise step-by-step reasoning with the knowledge.

Motivated by this, we introduce STRUCTCHEM, a sim-

¹“Formulae” and “formulas” are both correct plurals of “formula”, with “formulae” being preferred in scientific writing, per Garner’s Modern English Usage.

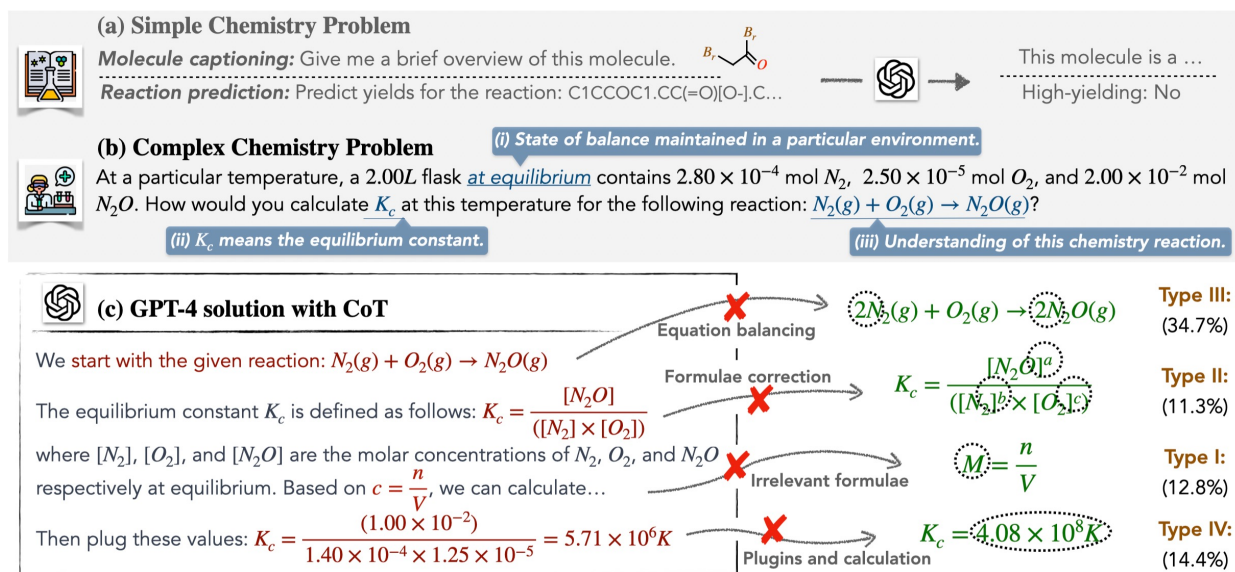


Figure 2. The illustration of (a) simple chemistry problem, (b) complex chemistry problem sampled from SciBench (Wang et al., 2023a), and (c) the zero-shot response from GPT-4 with chain-of-thought (CoT) (Wei et al., 2022) for the complex chemistry problem. The error types are illustrated corresponding to the definition in Figure 1: (I) irrelevant knowledge, (II) incorrect knowledge, (III) reasoning error, (IV) calculation error. We randomly select 100 error cases of GPT-4 (CoT) in SciBench.

ple yet effective reasoning strategy providing structured guidance for LLMs to solve complex chemistry problems. STRUCTCHEM explicitly decomposes the reasoning into three phases: In the *first* phase, the LLM focuses on generating essential chemical formulae needed for the problem. The formulae knowledge provides a solid basis for the LLM to do grounded reasoning in subsequent phases. The *second* phase involves the LLM conducting a detailed, step-by-step reasoning based on the identified formulae, leading to a preliminary answer to the problem. The *third* phase then performs *confidence-based* review-and-refinement for the final answer. Crucially, the refinement process differs from recent self-verification methods (Madaan et al., 2023; Weng et al., 2022) which rely solely on prompting and can sometimes yield unreliable results (Section 5). Instead, our approach explicitly estimates a confidence score for each revision, and iteratively enhances the confidence level towards a final high-quality answer.

We conduct extensive experiments on four datasets of complex chemistry problems from different subfields, namely, quantum chemistry, quantum mechanics, physical chemistry, and chemistry kinetics. Experiments show that STRUCTCHEM greatly reduces the reasoning errors (Figure 1). It boosts the chemistry reasoning capability of advanced LLMs, including GPT-3.5 and GPT-4, leading to an average improvement of 8% and a 30% absolute improvement at maximum. In addition, using the generated reasoning from our approach with GPT-4, we finetune smaller LMs (Llama-2-13B and Vicuna-13B) and obtain strong improve-

ment. This further validates STRUCTCHEM enables LLMs to generate high-quality chemistry reasoning. Our analysis studies the error patterns of LLMs on chemistry reasoning, which reveals the unique challenges in scientific problems and motivates future research towards more grounded and precise reasoning.

2. Related Work

2.1. Large Language Models for Chemistry

The emergence of LLMs has provided new possibilities in scientific domains (Ouyang et al., 2023), where a bunch of new benchmarks (Lu et al., 2022; Chen et al., 2023b) have emerged. As an important and challenging branch of scientific domains, chemistry-related research surges with the utilization of LLMs (Fang et al., 2023). Specifically, ChemCrow (Bran et al., 2023) is a general model that integrates multiple existing tools with LLMs to solve various downstream tasks. LLMs are also used to better the performance of specific chemistry applications, such as reaction prediction (Zhong et al., 2023; 2024), drug discovery (Edwards et al., 2023), and SMILES identification (Edwards et al., 2021). However, most previous works often target problems that require a single-hop retrieval of domain knowledge, without complex reasoning steps inherent. For example, find and list all the chemical entities in a given sentence. While the previous models excel in these approximate knowledge retrieval tasks, improving the abilities of LLMs to solve complex chemistry problems is still in a nascent stage, with

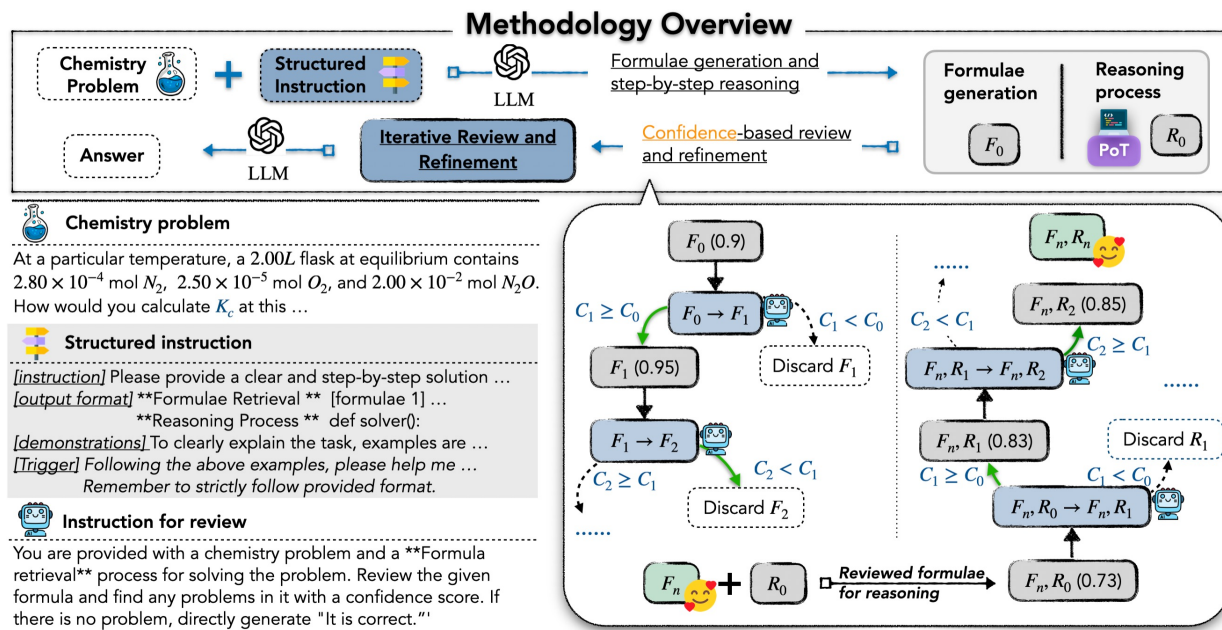


Figure 3. An in-depth illustration of the STRUCTCHEM framework. When tackling a chemistry problem, we first utilize a structured instruction approach, resulting in “formulae generation” \mathcal{F}_0 and “step-by-step reasoning” \mathcal{R}_0 . These generated segments are then fed to a thorough “confidence-based review-and-refinement” as initial input. The process is repeated n times til getting reviewed formulae \mathcal{F}_n and reasoning \mathcal{R}_n . Each iteration is guided by incorporating confidence scores C_i . \rightarrow in “iterative review and refinement” denote the choice made for each iteration. Full instructions can be found in Figure 14.

SciBench (Wang et al., 2023a) being the initial benchmark.

2.2. Large Language Models for Reasoning

Engaging LLMs in a step-by-step thinking process has demonstrated enhanced performance in intricate reasoning tasks compared to the conventional single-step answer prediction. A typical step-by-step prompting approach is called chain-of-thought (CoT) (Wei et al., 2022). CoT directs the model to articulate the step-by-step thinking process as rationales prior to producing the final answer. Following this line, several optimized endeavors are made towards making LLMs better task solvers for more complicated problems. These efforts encompass automated demonstration construction (Zhang et al., 2023), improving self-consistency (Wang et al., 2023b), utilizing the structure of prompts (Yao et al., 2023; Besta et al., 2024), adopting iterative prompting (Zhou et al., 2023; Wang et al., 2022), and ensembling (Sun et al., 2023; Fu et al., 2023). The original focus of CoT approaches has primarily centered around arithmetic, commonsense, and logical reasoning problems (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2023). Recent investigations have sought to broaden the application of CoT in scientific domains (Lu et al., 2022; Wang et al., 2023a). Closely related to ours are works in the research line of modular prompting (Khot et al., 2023; Patel et al., 2022), which decomposes a complex task into several sub-tasks. The micro-level

decomposition varies from task to task. Different from them, we identified two fundamental components for solving complex chemistry problems as a general paradigm at a macro level. Another related line is the feedback mechanism (Madaan et al., 2023) that leverages feedback from LLMs before the final output. In contrast, we design a confidence-based review-and-refinement strategy and employed another LLM to provide feedback for multi-model collaboration. Notably, this approach will greatly alleviate the drawbacks of previous feedback frameworks, where correct answers risk being swayed by unfaithful feedback.

3. STRUCTCHEM Reasoning

Solving complex chemistry problems not only necessitates recognizing domain knowledge, such as formulae and calculations but also demands the ability to construct a careful step-by-step reasoning process based on the relevant knowledge. Existing popular reasoning methods, such as CoT and self-consistency, though exhibit notable strengths, often fall short in accurately identifying the related chemistry formulae and are susceptible to errors in reasoning steps, as expounded in Section 1.

To address the challenges above, we propose STRUCTCHEM. On a high level, STRUCTCHEM consists of three stages: (i) formulae generation that offers the basis for subsequent

grounded reasoning; (ii) step-by-step reasoning that makes multi-step derivations with the identified formulae for a preliminary answer; and (iii) confidence-based review-and-refinement that steers LLMs to progressively revise the previous phases for increasing confidence, leading to the final high-confidence answer. Figure 3 the overall framework.

3.1. Formulae Generation

Formulae serve as organized and abstracted representations of chemistry knowledge (Lachmy et al., 2022). When humans tackle intricate problems, the initial phase often involves seeking relevant knowledge as a foundation, especially for the field of chemistry (Taskin & Bernholt, 2014). Therefore, rather than directly starting to address the question, we seek formulae to solve the problem first. Given the fact that LLMs have indeed encoded much chemistry knowledge, it is often effective to elicit the knowledge from the parametric storage (Petroni et al., 2019). Therefore, STRUCTCHEM first instructs the LLM to articulate relevant formulae for task resolution, exemplified by formulae like “equilibrium constant” in Figure 4. To enhance the utility of these formulae in subsequent reasoning processes, we instruct the LLM not only to recite them but also to provide explanations for the variables they contain. For instance, as illustrated in Figure 2, the LLM needs to elucidate *symbol* [*] as the molar concentrations.

3.2. Step-by-step Reasoning

Grounded on the generated formulae, the LLMs can then reason about the solution to the original question. To induce LLMs for more precise reasoning and calculation processes, we adopt program-of-thoughts (PoT) (Chen et al., 2023a) as demonstrated in Figure 5. The detailed calculation process is translated into Python codes, accompanied by the annotation lines for reasoning. Concretely, we feed the problem and the *structured instruction* into an LLM \mathcal{M} as shown in the top of Figure 3. The generated results of this stage are formalized as $S_0 = (\{\mathcal{F}_0, C_0^f\}, \{\mathcal{R}_0, C_0^r\})$ where $\mathcal{F}_0 = \{f_1, \dots, f_n\}$ denotes the formulae collected from Sec 3.1. f_n is the n -th formula that is related. Similarly, \mathcal{R}_0 denotes the reasoning process. The output samples of \mathcal{F}_0 and \mathcal{R}_0 could be found in Figure 4 and Figure 5. C_0^f and C_0^r are the confidence scores for each part that is going to be mentioned Sec 3.3.

3.3. Confidence-based Review-and-Refinement

The generated formulae and step-by-step reasoning are not always error-free. The cumulative errors in the formulae generation or step-by-step reasoning process can amplify and propagate throughout the entire generation, leading to wrong answers. Inspired by recent works of iterative prompting (Wang et al., 2022) and multi-model collaboration (Zheng et al., 2023), we employ the same LLM to conduct iteratively review-and-refine the previous iterations

Instruction for formulae generation

Firstly, for "formulae retrieval", you need to identify the formulae explicitly and implicitly entailed in the problem context, together with the necessary explanations for variables in the formulae.

Example

[Formula 1] $2N_2(g) + O_2(g) \rightarrow 2N_2O(g)$
 [Formula 2] $K_c = \frac{[N_2O]^a}{([N_2]^b \times [O_2]^c)}$, where a, b, c are the coefficients of different matters and [*] is the molarity of *.
 [Formula 3] $M = \frac{n}{V}$, where n is the mole, and V is the volume in terms of liters.

Figure 4. Instruction for formulae generation (part of the *structured instruction* in Figure 3) and the output for problem in Figure 2.

Instruction for reasoning process

Secondly, there is a "reasoning process" where you are required to reason step by step based on the identified formulae and problem context. Answer the question by implementing a solver function "def solver()" in Python.

Example

```
def solver():
    # Based on [Formulae 3], calculate the molarity of gases
    # entailed in the problem:
    N2O = 2.00/2.00 =1, O2 = 2.50/2.00=1.25, ...
    # Calculate for [Formulae 2]
    a = 1, b=2, c=1
    .....
```

Figure 5. Instruction for reasoning process (part of the *structured instruction* in Figure 3) and the output for problem in Figure 2.

of generation. In each iteration, the LLM is instructed to revise the formulae and reasoning steps from the previous iteration.

During the review process, we found that there are chances when correct answers are swayed by incorrect ones after revision. This phenomenon echoes the recent findings (Stechly et al., 2023; Huang et al., 2024) questioning the correction ability of LLMs existing in prevalent works such as self-consistency (Wang et al., 2023b) and self-verification (Weng et al., 2022). To fix this, we estimate a confidence score C_i on the revision process. Only a high-confidence revision is accepted for further refinement in the next iteration. The confidence assessment ensures each iteration makes meaningful progress towards final answers.

Specifically, we prompt the LLM to provide confidence scores C_i on a scale of $[0, 1]$ for the i -th iteration. The review process starts with the generated formulae $\{\mathcal{F}_0, C_0^f\}$, where C_0^f is the initial score for formulae generation. During the iterative review process, we leverage the same LLM to judge whether the collected formulae are correct with the confidence score C_i^f of its own prediction. Formulae with

Algorithm 1 Confidence-based Review-and-Refinement

```

Input:  $S_0 = (\mathcal{F}_0, \mathcal{R}_0)$ , initial confidence score  $C_0^f$  and  $C_0^r$ ,
LLM  $\mathcal{M}$ , prompts  $\{p_{rev}, p_{gen}\}$ , maximum iteration number  $n$ 
Output: Final answer to the problem  $P$ 
 $\mathcal{F}_0 = \{f_1, \dots, f_n\}$ ,  $\mathcal{R}_0 = \mathcal{R}_0$ 
// initialize maximum confidence scores
for iteration
 $max_f \leftarrow C_0^f$ ,  $max_r \leftarrow C_0^r$ 
// review for collected formulae
for  $i$  in  $1, \dots, n$  do
   $(\mathcal{F}_i, C_i^f) \leftarrow \mathcal{M}(p_{rev} || \mathcal{F}_{i-1})$ 
  if  $C_i^f < max_f$  then continue
   $\mathcal{F} = \mathcal{F}_i$ ,  $max_f \leftarrow C_i^f$ 
end for
// review for reasoning process
for  $i$  in  $1, \dots, n$  do
   $(\mathcal{R}_i, C_i^r) \leftarrow \mathcal{M}(p_{rev} || \mathcal{F} || \mathcal{R}_{i-1})$ 
  if  $C_i^r < max_r$  then continue
   $\mathcal{R} = \mathcal{R}_i$ ,  $max_r \leftarrow C_i^r$ 
end for
return  $\mathcal{M}(p_{gen} || \mathcal{F} || \mathcal{R})$ 

```

the highest confidence score are kept. We then repeat it for the reasoning process so that we get the most faithful generations for both parts. In this way, LLMs can select the most aligned combination of problem, formulae, and the reasoning process. The elements above are finally input into the same LLM to get the final answer. The overall pipeline of this stage is shown in Algorithm 1, where the prompts $\{p_{rev}, p_{gen}\}$ denote the instruction for “review” and “generation” with refined results.

4. Experiments

4.1. Setup

In our experiments, we use four datasets taken from SciBench (Wang et al., 2023a). The datasets cover a wide range of subfields including quantum chemistry, physical chemistry, kinetics, and matter, etc. The detailed distribution of subfields is shown in Figure 9. The four datasets are manually collected from college-level chemistry textbooks, and are selected to be more challenging with free-response answers. Each of the datasets is divided into two parts, \mathcal{P}_w and \mathcal{P}_s . Here \mathcal{P}_w contains the majority number of problems that without solutions. Meanwhile, problems in \mathcal{P}_s are coupled with solutions. The complexity of these datasets could also be proved by the average number of formulae entailed (around 2) and the average reasoning steps (around 5) generated by STRUCTCHEM needed to solve the problems. (Detailed distribution information could be found in Appendix B.)

Experiments are conducted under both zero-shot and few-shot settings. For the few-shot setting, the demonstrations are constructed with 3 examples randomly sampled from

\mathcal{P}_s . We leverage GPT-3.5 (gpt-3.5-turbo) and GPT-4 (gpt-4-0315) as our backbone models. For each setting, we consider four baselines following the evaluation paradigm in SciBench (Full instructions are provided in Appendix A):

- (i) **Direct reasoning** refers to directly feeding the problem into the model without any other instructions;
- (ii) **System instruction** is originally developed by Wang et al. (2023a) which is tailored to the task and describes the types and categories of questions, along with instructions;
- (iii) **CoT** follows the “step-by-step” prompting strategy that requires the model to output the “thinking process” first;
- (iv) **CoT-SC** is an improved version of CoT with self-consistency (Wang et al., 2023b). It adds more rounds of CoT reasoning and outputs the most consistent results.
- (v) **ToT** instructs the LLM to strategically explore (Yao et al., 2023), considering multiple different reasoning paths in a tree structure.

(vi) **PoT** leverages the idea of Program-of-Thoughts (PoT) (Chen et al., 2023a) which translates the solution into Python codes to improve the understanding and calculation ability of LLMs.

4.2. Implementation Details

We access the two LLMs, GPT-3.5 and GPT-4, with the OpenAI API. During our experiments, the temperature for generation is kept at 0 to ensure reproducibility and reduce potential variances. For baseline implementations, we follow the default settings in the corresponding papers. Specifically for CoT-SC, we set the temperature to 0.7, and the batch size of rationale to 20. For ToT, we set the temperature to 1.0, with “n_generate_samples” and “n_evaluate_samples” setting to 5. When doing the evaluation, we follow the previous work (Wang et al., 2023a) and compare the model outputs with the correct answers, allowing an absolute deviation of 0.1 for answers greater than 1 and a relative tolerance of 0.05 for answers less than 1. This guarantees a fair comparison with previous baseline models.

4.3. Results

Table 1 and 4 present the performance of all methods on the test set of the four datasets. We report the model performance in terms of accuracy scores. The best results are bolded. Based on the results, we have the following key observations:

- (i) **STRUCTCHEM achieves superior performance on almost all the datasets in both zero-shot and few-shot settings.** Specifically, STRUCTCHEM achieves an absolute improvement of +13.77 and +7.71 in terms of the average score on few-shot settings, respectively, which is

Table 1. Results on the test sets for the four datasets, *quan*, *chemmc*, *atkins*, *matter*. We compare with baselines from two different settings, a zero-shot setting with no demonstrations and a few-shot setting with 3 demonstrations. We compute the accuracy scores with the approximation detailed in Section 4.3. The best results for each setting are highlighted in **bold** and the second-best results are underlined.

Methods	GPT-3.5				Avg.	GPT-4				Avg.
	<i>quan</i>	<i>chemmc</i>	<i>atkins</i>	<i>matter</i>		<i>quan</i>	<i>chemmc</i>	<i>atkins</i>	<i>matter</i>	
Zero-shot setting										
Direct Reasoning	5.88	28.21	<u>8.41</u>	4.08	11.65	8.82	25.64	14.95	18.37	16.95
System Instruction	8.82	20.51	<u>4.67</u>	2.04	9.01	14.71	23.08	27.10	<u>22.45</u>	21.84
CoT	2.94	<u>23.08</u>	6.54	<u>10.20</u>	10.69	14.71	43.59	<u>28.04</u>	<u>20.41</u>	<u>26.69</u>
PoT	0.0	<u>7.69</u>	0.0	<u>2.04</u>	2.43	11.76	20.51	25.23	16.33	18.46
STRUCTCHEM	<u>5.88</u>	15.38	9.35	12.24	<u>10.71</u>	20.59	<u>38.46</u>	31.78	24.49	30.11
Few-shot setting										
Direct Reasoning	5.88	23.08	9.35	8.16	11.62	14.71	28.21	20.69	14.29	19.48
System Instruction	11.76	15.38	5.61	4.08	9.21	17.65	30.77	15.87	12.24	19.13
CoT	8.82	20.51	8.41	6.12	10.97	17.65	46.15	21.05	26.53	27.85
CoT-SC	<u>14.71</u>	25.64	<u>15.89</u>	16.33	<u>18.14</u>	29.41	<u>51.28</u>	<u>50.47</u>	<u>28.57</u>	<u>39.93</u>
ToT	11.76	17.95	11.21	14.29	13.80	23.53	43.59	32.71	20.41	30.06
PoT	8.82	<u>33.33</u>	13.08	16.33	17.89	<u>38.24</u>	41.03	21.05	<u>28.57</u>	32.22
STRUCTCHEM	32.35	43.59	26.17	24.49	31.66	41.18	58.97	59.81	30.61	47.64

Table 2. Detailed statistics and information of the four datasets we experiment with. $\#P_w$ and $\#P_s$ refer to the number of data samples with and without solutions. “# F” means the average number of formulae entailed in the problem, and “# RS” denotes the average reasoning steps for each problem.

Datasets	Subfields/Topics	$\#P_w(P_s)$	# F	# RS
<i>quan</i>	Quantum chemistry	34 (8)	1.93	3.94
<i>chemmc</i>	Quantum mechanics	39 (9)	1.88	3.95
<i>atkins</i>	Physical chemistry	107 (16)	1.65	4.33
<i>matter</i>	Chemistry kinetics	49 (10)	1.89	4.43

43.49%, and 19.31% of relative improvement. The notable improvement demonstrates the effectiveness of our model in adapting to various scenarios by inducing chemistry knowledge and performing precise reasoning. The improvement brought by few-shot learning is also larger than the zero-shot setting. Compared with the best general prompt strategies, CoT-SC and ToT, STRUCTCHEM is still the best. Notably, ToT does not meet expectations, as it excels at solving problems requiring many reasoning steps but struggles with ones of this complexity.

(ii) **STRUCTCHEM has shown effectiveness generally across different backbones.** In addition, using GPT-4 as backbone models consistently outperforms GPT-3.5 backbones for all methods experimented by a large margin. Specifically, we found that STRUCTCHEM achieves more pronounced performance improvement with GPT-4 under the zero-shot setting, with +3.42 of accuracy improvement. However, when it comes to the few-shot setting, STRUCTCHEM works better with GPT-3.5. Given the fact that GPT-4 is a more powerful model than GPT-3.5, this result actually indicates a nuanced balance in terms of instruction-following ability and demonstrations. When

there is no demonstration available, the performance is more dependent on the inherent power of LLMs, where GPT-4 prevails. With enough demonstrations and detailed prompts, even weaker LLMs as GPT-3.5 could be powerful with STRUCTCHEM.

(iii) **STRUCTCHEM delivers greater performance improvements on four datasets in few-shot settings compared to zero-shot settings.** With GPT-4, the performance improvement brought by STRUCTCHEM is +12.0 more than the zero-shot setting. The reason could be twofold. One is the complexity of these datasets, because it is hard to directly output answers or solutions without any demonstrations as references. Another factor could be the format of solutions offered by STRUCTCHEM, which disentangles the solution with structured instructions. This format is hard to learn with no examples. Contrastively, we do not observe obvious differences in performance improvement brought by baselines for few-shot and zero-shot settings. This further shows STRUCTCHEM’s ability to learn by analogy.

(iv) **STRUCTCHEM achieves substantial performance gains in complex problems with extensive reasoning steps.** Specifically, we found the performance on datasets *chemmc* and *atkins* is better than *quan* and *matter* with GPT-4 in few-shot setting. STRUCTCHEM performs particularly well on *atkins* dataset, with the accuracy scores doubling in almost all settings. We attribute the reason to the number of formulae in *atkins*. As shown in Table 2, *atkins* has the smallest average number of formulae, making it easier for formulae collection. However, it has a larger number of average reasoning steps. This verifies STRUCTCHEM’s ability to deal with complicated reasoning processes.

(v) **STRUCTCHEM is well instantiated with open-source LLMs.** We also tested STRUCTCHEM with open-source

Table 3. Ablation studies for different components in STRUCTCHEM in both zero-shot and few-shot settings with the backbone model GPT-4. The accuracy scores are reported with all four datasets.

Methods	Zero-shot				Avg.	Few-shot				Avg.
	<i>quan</i>	<i>chemmc</i>	<i>atkins</i>	<i>matter</i>		<i>quan</i>	<i>chemmc</i>	<i>atkins</i>	<i>matter</i>	
structured instruction	23.53	35.90	39.25	18.37	29.26	32.35	51.28	53.27	28.57	41.37
+ review for \mathcal{F}	<u>26.47</u>	38.46	40.19	18.37	30.87	32.35	48.71	54.55	30.61	41.56
+ review for \mathcal{R}	23.53	41.03	<u>41.12</u>	22.45	<u>32.03</u>	35.29	51.28	54.55	30.61	42.93
+ confidence score	29.41	41.03	46.34	<u>23.08</u>	34.97	38.24	53.85	56.07	32.65	45.20
+ PoT	20.59	<u>38.46</u>	31.78	24.49	30.11	41.18	58.97	59.81	30.61	47.64

Table 4. Results of STRUCTCHEM with open-source LLMs, including Llama-2-7B and Llama-2-70B under the few-shot setting.

Methods	<i>quan</i>	<i>chemmc</i>	<i>atkins</i>	<i>matter</i>	Avg.
w/ Llama-2-7B					
CoT	2.94	5.13	1.87	0.00	2.49
PoT	0.00	2.56	0.93	0.00	0.87
STRUCTCHEM	2.94	7.69	8.41	2.04	5.27
w/ Llama-2-70B					
CoT	14.71	12.83	13.10	4.08	11.18
PoT	2.94	7.69	0.93	0.00	2.89
STRUCTCHEM	17.65	17.95	18.69	6.12	15.10

LLMs, specifically the Llama-2 series (Touvron et al., 2023), with results shown in Table 4. We found that STRUCTCHEM still outperforms strong baselines with both model scales. However, STRUCTCHEM does not perform that well with LLMs in smaller sizes such as Llama-2-7B. Potential reasons include the relatively weak instruction-following ability of these models and the complexity of the task itself.

5. Analysis

Intuitively, we want to validate the quality of produced chemistry reasoning processes. We first fine-tune smaller models using generated reasoning outputs. We then conduct a thorough ablation study of STRUCTCHEM’s various components to gain a deeper understanding of its effectiveness. Additionally, an error analysis further offers insights about how to make STRUCTCHEM even better in the future.

5.1. Validating the Reasoning Quality

Our method STRUCTCHEM has shown strong improvement in accuracy over baselines. Here we further validate that STRUCTCHEM generates high-quality intermediate reasoning steps that increase answer accuracy. Specifically, we fine-tune smaller language models, Llama-2-13B-chat (Touvron et al., 2023) and Vicuna-13B (Chiang et al., 2023), on the reasoning steps generated by STRUCTCHEM and CoT, respectively. The rationale is that while a smaller model may already be equipped with some domain knowledge, it typically lacks the capability for step-by-step reasoning in complex chemistry problems—a skill that emerges pre-

dominantly in larger-scale models. By fine-tuning smaller models with the generated reasoning steps, we essentially teach them to perform this advanced reasoning. Intuitively, using higher-quality fine-tuning data would lead to better performance in the small models.

To collect fine-tuning data, we first instruct GPT-4 to generate another 1,000 problems with 3 problems sampled from SciBench as demonstrations. To encourage diversity, we set the generation temperature as 1.0 and filter out problems that have 5-gram or larger overlapping with existing generated problems. Then, we use STRUCTCHEM for providing the solutions to all 1,000 problems as the paired training data. Additionally, we compare two other baselines: (i) Fine-tuning model on the original data, which only consists of the original problem statement and the direct answer, formatted as “[problem] The answer is therefore [answer]”; (ii) Pure zero-shot inference, where given the problem as input, the model outputs a direct answer without any fine-tuning.

The fine-tuning process is based on LoRA (Hu et al., 2022), a parameter-efficient fine-tuning method. For details on training and problem generation, please refer to Appendix C.

Results are shown in Table 5. The vanilla version falls short in solving such complex chemistry problems, as shown by their zero-shot performance on four datasets. Training only on the original problem and answer pairs does not bring much improvement compared with direct inference. Fine-tuning with data generated by STRUCTCHEM, on the other hand, brings more than 20% absolute improvement. Fine-tuning based on STRUCTCHEM is superior to fine-tuning with CoT, demonstrating that STRUCTCHEM can produce detailed reasoning at a higher quality.

5.2. Ablation Study

To understand why STRUCTCHEM works particularly well, we ablate STRUCTCHEM with add-ons from different components. Table 3 summarizes the experimental results.

Firstly, both “structured instruction” and “iterative review and refinement” are significant in contributing to the performance of STRUCTCHEM for zero-shot and few-shot settings. Specifically, removing the confidence score and iterative review resulted in a decrease of 2.27 and 3.83, respectively.

Table 5. Fine-tuning results with generations from STRUCTCHEM as training data on two open-source models. The accuracy scores are reported with all four datasets.

Methods	Llama-2-13B-chat				Avg.	Vicuna-13B				Avg.
	quan	chemmc	atkins	matter		quan	chemmc	atkins	matter	
Zero-shot inference	0.0	0.0	0.0	2.04	0.51	5.88	2.56	3.74	0.0	3.05
Original	0.0	5.13	0.0	0.0	1.28	8.82	5.13	2.80	2.04	4.70
CoT	8.82	17.95	9.35	8.16	11.07	11.76	15.38	8.41	6.12	10.42
STRUCTCHEM	14.71	30.77	20.56	20.41	21.61	17.65	33.33	18.69	20.41	22.52

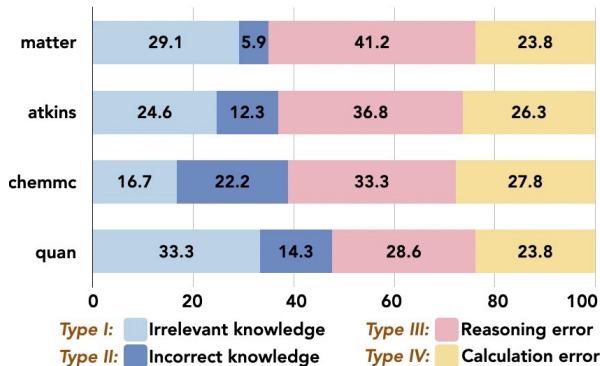


Figure 6. Error analysis of four error categories across all datasets (y-axis) in terms of error proportions (x-axis). The results are for STRUCTCHEM *w/o* PoT to exclude the influence of external tools.

It is worth noting that while iterative refinement indeed contributes to the performance, our strategy of structured instruction is strong enough and demonstrates comparative performance with strong baselines such as CoT. When removing iterative review for formulae \mathcal{F} alone, the performance drops by a large margin, which is comparable to removing the whole iterative review process. This shows the effectiveness of iterative review for formulae collection and the importance of domain knowledge when solving.

Also, though PoT helps with precise calculation and improves performance, STRUCTCHEM without PoT still outperforms the strongest baselines. We also note that in the zero-shot setting, STRUCTCHEM *without* PoT achieves even stronger performance. Although PoT indeed helps with the accuracy of calculations, it does not make up for the disadvantage of instruction-following ability in the zero-shot setting. By adding a few demonstrations for few-shot PoT, this disadvantage was made up for.

5.3. Error Analysis

We conduct a manual analysis of all the 113 error cases for STRUCTCHEM *without* PoT with GPT-4 as the backbone for few-shot setting across four datasets. Error types are defined as corresponding to the two processes of “formulae generation” and “step-by-step reasoning”. The analysis is done by 3 Ph.D. students with a chemistry background.

For “formulae generation”, we define two types of errors that are related to this process. **Irrelevant knowledge** indicates that the formulae collected are not relevant to solving the problem. For example, solving a problem requires *Broglie formula* but LLM collects *Wavelength formula*. **Incorrect knowledge** refers to the incorrectness inherent in the formula itself. $K_c = \frac{[N_2O]}{[N_2] \times [O_2]}$ in Figure 2 is one such example. For “step-by-step reasoning”, we also have two error cases as follows. **Reasoning error** refers to the errors made during the intermediate reasoning steps. For example, in Figure 2, the model fails to reason the correct relations of different gases during the reaction $O_2 + N_2 \rightarrow N_2O$. **Calculation error** means the mathematical computation mistakes made when doing the reasoning process.

The results are shown in Figure 6, where we plot the proportion for every error type of each dataset. We have the following key observations:

- (i) **STRUCTCHEM are more likely to generate irrelevant formulae than inaccurate ones.** On average, only 13.7% of the total errors are caused by incorrect forms of formulae compared with an average of 25.9% of irrelevant ones. The irrelevance rate is slightly higher than that of GPT-4 (CoT) as shown in Figure 1. A potential reason is that STRUCTCHEM could focus on the irrelevant formulae collected in the first phase. For the entire formulae collection process, although STRUCTCHEM sometimes retrieve irrelevant formulae for solving a problem, the formulae are less likely to be incorrect themselves.
- (ii) **Formulae being relevant probably is more important than being correct.** We observed that the “Irrelevance” rate is relatively low for *atkins* and *chemmc* datasets, although they may have higher “incorrectness” rate. This is potentially another explanation of why performance in Table 1 is particularly high for these two datasets compared with the rest two datasets, since the formulae collection process serves as a necessary condition for conducting correct reasoning processes further.
- (iii) **Complex reasoning ability is still the bottleneck of LLMs.** Although STRUCTCHEM drastically decreases the reasoning error as shown in Figure 1, “reasoning error” still takes up to around 35.0% of all error cases. For chemistry

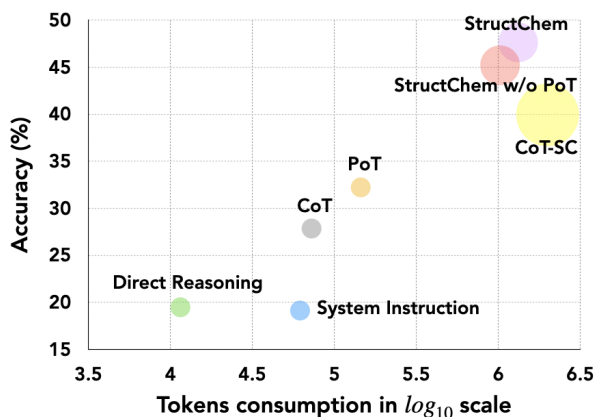


Figure 7. Cost-effectiveness analysis. The size of each dot is proportional to the average number of inferences by each method. The y-axis denotes the average accuracy across four datasets.

problems that entail multiple elements interacting in a complex environment, the ability to reason out the relations among objects becomes crucial.

(iv) **Preciseness is important for solving complex chemistry problems.** Without PoT, the case of “calculation error” still occupies a large portion of around a quarter. Even a single step of calculation error could lead to wrong answers in chemistry reasoning problems.

5.4. Cost-Effectiveness Analysis

By introducing STRUCTCHEM, we manage to reduce the costs associated with complex chemistry problems while achieving comparable or even superior performance. We conduct experiments in the few-shot setting with GPT-4 as the backbone. We define cost as the sum of tokens for instruction, demonstrations, and output. Based on results illustrated in Figure 7, we can see that the performance increase brought by STRUCTCHEM is actually a little larger compared to CoT and PoT considering the ratio of tokens consumption. CoT-SC, while achieving the most competitive results among baselines, actually consumes more tokens and requires around twice the amount of function calls. STRUCTCHEM’s substantial improvement does not rely on the consumption of tokens.

6. Conclusion and Discussion

This paper introduces STRUCTCHEM, a new reasoning structure that guides LLMs to solve complex chemistry problems. STRUCTCHEM explicitly decomposes the reasoning into three critical phrases, including *formulae generation* by LLMs that offers the basis for grounded reasoning, *step-by-step reasoning* that makes derivations with the identified formulae for a preliminary answer, and *confidence-based review-and-refinement* that steers LLMs to progressively re-

view the previous phases, leading to the final high-confidence answer. Extensive experiments on four datasets of complex chemistry problems from different subfields of chemistry show that STRUCTCHEM significantly boosts the chemistry reasoning capability of different LLMs. In addition, we fine-tune smaller LMs (e.g., Vicuna-13B) using the generated reasoning from our approach with GPT-4 and obtain strong improvement. Future work could continue to investigate incorporating external, up-to-date knowledge sources and performing retrieval to ensure the quality of the formulae generation. Or designing strategies to transfer and distill chemistry reasoning knowledge from LLMs to smaller LMs.

Acknowledgements

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004, National Science Foundation IIS-19-56151, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA, the National Science Foundation, or the U.S. Government.

Impact Statement

The impact of this paper lies in its significant contribution to improving the ability of Large Language Models (LLMs), like GPT-4, in the domain of complex chemistry reasoning. The introduction of STRUCTCHEM not only elevates the LLMs’ performance in chemical reasoning tasks but also demonstrates the potential to integrate with other reasoning tools and strategies, thereby pushing the boundaries of what artificial intelligence can achieve in the realm of scientific inquiry. This advancement opens up new avenues for using LLMs in scientific research, education, and industry, where they can assist in acting as teaching agents, conducting experiments, or even in developing new materials and drugs. Furthermore, by highlighting the ongoing challenges and the need for further research in precise, grounded reasoning with LLMs, the paper sets a new direction for future work in AI and machine learning, encouraging a deeper investigation into how AI can more effectively mimic human-like reasoning in scientific domains. Overall, we do not foresee any major risks or negative societal impacts of our work. All the datasets we experiment with are publicly available online. We followed the licenses when conducting experiments on publicly available datasets and human annotations. We will open-source this project upon acceptance to facilitate future research, especially for small research groups or institutions with relatively fewer resources of LLMs.

References

- Atkins, P., De Paula, J., and Friedman, R. *Physical chemistry: quanta, matter, and change*. Oxford University Press, USA, 2014.
- Atkins, P., De Paula, J., and Keeler, J. *Atkins' physical chemistry*. Oxford university press, 2023.
- Baum, Z. J., Yu, X., Ayala, P. Y., Zhao, Y., Watkins, S. P., and Zhou, Q. Artificial intelligence in chemistry: current trends and future directions. *Journal of Chemical Information and Modeling*, 61(7):3197–3212, 2021.
- Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., and Hoefler, T. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16): 17682–17690, Mar 2024. doi: 10.1609/aaai.v38i16.29720. URL <https://ojs.aaai.org/index.php/AAAI/article/view/29720>.
- Bran, A. M., Cox, S., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.
- Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856. URL <https://openreview.net/forum?id=YfZ4ZPt8zd>.
- Chen, W., Yin, M., Ku, M., Lu, P., Wan, Y., Ma, X., Xu, J., Wang, X., and Xia, T. TheoremQA: A theorem-driven question answering dataset. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.489. URL <https://aclanthology.org/2023.emnlp-main.489>.
- Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Edwards, C., Zhai, C., and Ji, H. Text2Mol: Cross-modal molecule retrieval with natural language queries. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 595–607, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.47. URL <https://aclanthology.org/2021.emnlp-main.47>.
- Edwards, C., Lai, T., Ros, K., Honke, G., Cho, K., and Ji, H. Translation between molecules and natural language. *arXiv preprint arXiv:2204.11817*, 2022.
- Edwards, C. N., Naik, A., Khot, T., Burke, M. D., Ji, H., and Hope, T. Synergpt: In-context learning for personalized drug synergy prediction and drug design. *bioRxiv*, pp. 2023–07, 2023.
- Fang, Y., Liang, X., Zhang, N., Liu, K., Huang, R., Chen, Z., Fan, X., and Chen, H. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. *arXiv preprint arXiv:2306.08018*, 2023.
- Feinberg, E. N., Sur, D., Wu, Z., Husic, B. E., Mai, H., Li, Y., Sun, S., Yang, J., Ramsundar, B., and Pande, V. S. Potentialnet for molecular property prediction. *ACS central science*, 4(11):1520–1530, 2018.
- Fu, Y., Peng, H., Sabharwal, A., Clark, P., and Khot, T. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=yflicZHC-19>.
- Graulich, N., Rost, M., Schultz, M., and Gallardo-Williams, M. To read is the challenge - insights from 100 days, 100 papers reading challenge in chemistry education research. *Journal of Chemical Education*, 99(10):3364–3369, 2022.
- Hair, J., Black, W., Babin, B., Anderson, R., and Tatham, R. Pearson prentice-hall; upper saddle river, nj. *Multivariate data analysis.*[Google Scholar], 2009.
- Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ik_mD3fKBPQ.
- Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda*,

- May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/pdf?id=_nGgzQjzaRy.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.
- Lachmy, R., Pyatkin, V., Manevich, A., and Tsarfaty, R. Draw me a flower: Processing and grounding abstraction in natural language. *Transactions of the Association for Computational Linguistics*, 10:1341–1356, 2022. doi: 10.1162/tacl.a.00522. URL <https://aclanthology.org/2022.tacl-1.77>.
- Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoye, S., Yang, Y., Welleck, S., Majumder, B. P., Gupta, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651, 2023. doi: 10.48550/ARXIV.2303.17651. URL <https://doi.org/10.48550/arXiv.2303.17651>.
- McQuarrie, D. A. *Quantum chemistry*. University Science Books, 2008.
- Ouyang, S., Wang, S., Liu, Y., Zhong, M., Jiao, Y., Iter, D., Pryzant, R., Zhu, C., Ji, H., and Han, J. The shifted and the overlooked: A task-oriented investigation of user-GPT interactions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=qSlip2dGH0>.
- Öztürk, H., Özgür, A., Schwaller, P., Laino, T., and Ozkirimli, E. Exploring chemical space using natural language processing methodologies for drug discovery. *Drug Discovery Today*, 25(4):689–705, 2020.
- Patel, P., Mishra, S., Parmar, M., and Baral, C. Is a question decomposition unit all we need? In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4553–4569, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.302. URL <https://aclanthology.org/2022.emnlp-main.302>.
- Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., and Miller, A. Language models as knowledge bases? In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL <https://aclanthology.org/D19-1250>.
- Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.
- Stechly, K., Marquez, M., and Kambhampati, S. Gpt-4 doesn’t know it’s wrong: An analysis of iterative prompting for reasoning problems. *arXiv preprint arXiv:2310.12397*, 2023.
- Sun, Z., Wang, X., Tay, Y., Yang, Y., and Zhou, D. Recitation-augmented language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=-cqvvvb-NkI>.
- Taskin, V. and Bernholt, S. Students’ understanding of chemical formulae: A review of empirical research. *International Journal of Science Education*, 36(1):157–185, 2014.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Wang, B., Deng, X., and Sun, H. Iteratively prompt pre-trained language models for chain of thought. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2714–2730, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.174. URL <https://aclanthology.org/2022.emnlp-main.174>.
- Wang, X., Hu, Z., Lu, P., Zhu, Y., Zhang, J., Subramaniam, S., Loomba, A. R., Zhang, S., Sun, Y., and Wang, W. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023a.
- Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=1PL1NIMMrw>.

- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J. 2023. URL <https://openreview.net/forum?id=WZH7099tgfM>.
- Weng, Y., Zhu, M., He, S., Liu, K., and Zhao, J. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*, 2022.
- Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=5Xc1ecx0lh>.
- Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023.
- Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685, 2023. doi: 10.48550/ARXIV.2306.05685. URL <https://doi.org/10.48550/arXiv.2306.05685>.
- Zhong, M., Ouyang, S., Jiang, M., Hu, V., Jiao, Y., Wang, X., and Han, J. ReactIE: Enhancing chemical reaction extraction with weak supervision. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 12120–12130, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.767. URL <https://aclanthology.org/2023.findings-acl.767>.
- Zhong, X., Du, Y., Ouyang, S., Zhong, M., Luo, T., Ho, Q., Peng, H., Ji, H., and Han, J. Actionie: Action extraction from scientific literature with programming languages. In *the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, August 2024.
- Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*,

A. Prompts used for baseline methods in Section 4.

In this section, we provide the detailed prompts used for experiments.

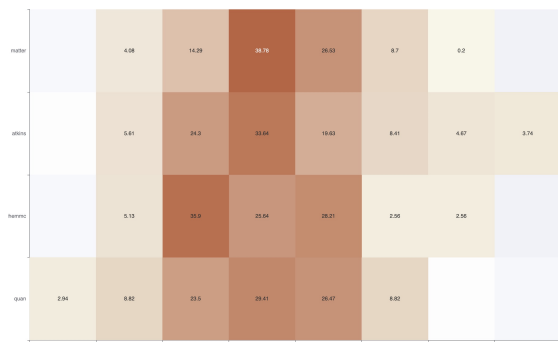


Figure 8. Distribution of four datasets in terms of the reasoning steps.

System Prompt Please provide a clear and step-by-step solution for a scientific problem in the category of Chemistry. The problem will specify the unit of measurement, which should not be included in the answer. Express the final answer as a decimal number with three digits after the decimal point. Conclude the answer by stating “The answer is therefore [ANSWER].”

Program-of-Thought Prompt Please provide a clear and step-by-step solution for a scientific problem in the category of Chemistry. The problem will specify the unit of measurement. Please translate the solution steps into Python code and encase the Python code within triple backticks for clarity.

Template-guided Prompt The full prompt for “formulae generation” and “step-by-step reasoning” is composed of four stages, general instruction, output format, demonstrations, and trigger. The complete view of the prompt is shown in Figure 14.

B. Distribution of datasets

The detailed distribution of four datasets in terms of reasoning steps is shown in Fig 8. We can see that the majority of the samples have reasoning steps spanning [3, 5]. Some samples even have reasoning steps of 8, which demonstrate the complexity of these datasets.

C. Details for Section 5.3

Instruction for problem generation Please help me to generate complex and difficult chemistry problems that include but are not limited to the fields of physical chemistry,

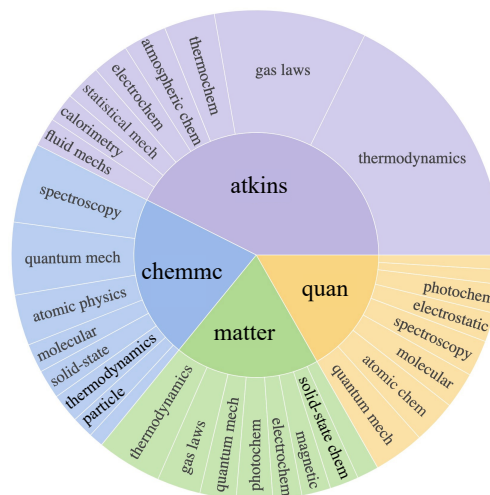


Figure 9. *Quantum chemistry (quan)* (Hair et al., 2009) provides an exploration of equilibrium, structure, and reactions. *Chemistry kinetics (matter)* (Atkins et al., 2014) combines physics and mathematics, spanning through quantum mechanics and atomic structure. *Quantum mechanics (chemmc)* (McQuarrie, 2008) covers quantum mechanics and the applications in chemical bonding. *Physical chemistry (atkins)* (Atkins et al., 2023) provides explorations of equilibrium, structure, and reactions. We leverage GPT-4 to annotate each data sample in these datasets for the specific subfields.

quantum chemistry, thermodynamics, atomic chemistry, molecular, etc. To help you better understand, I provide the following examples: [demonstrations]. Following the above examples, please help me with this task and generate three problems that satisfy my requirements. Make sure the generated problems are reasonable and complex for solving.

Training details We use LLaMA-2-13B-chat (Touvron et al., 2023) and Vicuna-13B-v1.3 (Chiang et al., 2023) as backbone models and finetune them with the LoRA approach (Hu et al., 2022). During training, we configure the batch size to 8 and the maximum learning rate to $1e-4$ with a 0.03 warmup ratio. For all the experiments, the LoRA r is set to 8, and we apply a dropout rate of 0.05. We keep these hyperparameters the same for a fair comparison. We train the models with 10 epochs and it takes around 1 hour to train on a single NVIDIA A6000 GPU. During the inference process, we also adhere to the same set of parameters: a temperature of 0.1, top-p of 0.75, top-k of 40, 4 beams, and a maximum generation length of 2,048.

D. Examples of error type in Section 5.2

To help better understand the error category listed in Section 5.3, we provide one example for each category in Figure 12, Figure 13.

Problem	The $J = 2 \rightarrow 3$ rotational transition in a certain diatomic molecule occurs at 126.4 GHz, where 1 GHz = 10^9 Hz. Find the frequency of the $J = 5 \rightarrow 6$ absorption in this molecule.
Answer	<p>Formula retrieval: [Formula 1] $E = 2B(J + 1)$, where B is the rotational constant, ... [Formula 2] $E = h\nu$, where h is the Planck's constant, ...</p> <p>Reasoning: [Step 1] Calculate the energy difference for transition from 2 to 3. $\Delta E_{2 \rightarrow 3} = h\nu_{2 \rightarrow 3} = 6.626 \times 10^{-34} \text{ J s} \cdot 126.4 \times 10^9 \text{ Hz}$</p> <p>[Step 2] Calculate the rotational constant B. Using energy difference and initial rotational quantum number $\Delta E_{2 \rightarrow 3} = 2B(2 + 1)$, hence $B = \frac{\Delta E_{2 \rightarrow 3}}{6}$</p> <p>[Step 3] Calculate the energy difference for transition from 5 to 6 $\Delta E_{5 \rightarrow 6} = 2B(5 + 1)$</p> <p>[Step 4] ...</p>
Correct	252.8 V

Figure 10. An example of a reasoning error. The red highlighted expression is deduced from the given formula. Instead of solving for difference, it directly plug the value for calculating energy. The correct expression for the first one should be $\Delta E_{2 \rightarrow 3} = 2B(3 + 1) - 2B(2 + 1)$.

Problem	Take nitrogen to be a van der Waals gas with $a = 1.352 \text{ dm}^6 \text{ atm mol}^{-2}$ and $b = 0.0387 \text{ dm}^3 \text{ mol}^{-1}$, and calculate ΔH_m when the pressure on the gas is decreased from 500 atm to 1.00 atm at 300 K. For a van der Waals gas, $\mu = \left(\frac{2a}{RT} - b\right)/C_{p,m}$. Assume $C_{p,m} = \frac{7}{2}R$.
Answer	<p>Formula retrieval: [Formula 1] $\mu = \left(\frac{2a}{RT} - b\right)/C_{p,m}$, given formula for a van der ... [Formula 2] $\Delta H_m = \int \mu dP$, general formula for calculating change in enthalpy with [Formula 3] ...</p> <p>Reasoning: [Step 1] ...</p>
Correct	+3.59 KJ/mol

Figure 11. An example of a factual error. The red highlighted expression is the wrong form to calculate. The correct formulae should be $\int_p^{p'} dH_m$, which could be further converted to $H_m q x = H_m(v' - d)$.

Problem	Through what potential must a proton initially at rest fall so that its de Broglie wavelength is $1.0 \times 10^{-10} \text{ m}$?
Answer	<p>Formula retrieval: [Formula 1] $\lambda = h/p$, where λ is the de Broglie wavelength, ... [Formula 2] $K = qV$, where k is the kinetic energy, ... [Formula 3] $K = 1/2mv^2$, where m is the mass of the particle, ...</p> <p>Reasoning: [Step 1] Calculate the momentum of the proton using the de Broglie wavelength formula. $p = \frac{h}{\lambda} = 6.626 \times 10^{-34} \times 1.0 \times 10^{-10} = 6.626 \times 10^{-24} \text{ kg m s}^{-1}$ [Step 2] Calculate the kinetic energy of the proton using its momentum. $K = \frac{p^2}{2m} = \frac{(6.626 \times 10^{-24})^2}{2(1.673 \times 10^{-27})} = 1.381 \times 10^{-17} \text{ J}$ [Step 3] Calculate the potential difference using the kinetic energy formula. $V = \frac{K}{q} = 1.381 \times \frac{10^{-17}}{1.602} \times 10^{-19} = 86.2 \text{ V}$</p>
Correct	0.082 V

Figure 12. An example of a calculation error. The red highlighted expression is wrongly calculated. If calculated correctly, we will get $K = 1.312 \times 10^{-20}$, which yields the correct answer.

Problem	Use the D_0 value of H_2 (4.478 eV) and the D_0 value of H_2^+ (2.651 eV) to calculate the first ionization energy of H_2 (that is, the energy needed to remove an electron from H_2). The unit is eV.
Answer	<p>Formula retrieval: [Formula 1] $E_1 = D_0(H_2) - D_0(H_2^+)$, where E_1 is the first ionization energy, ...</p> <p>Reasoning: [Step 1] Substitute the given values into the formulae and calculate the first ionization energy $E_1 = 1.827 \text{ eV}$</p>
Correct	15.43 eV

Figure 13. An example of a principle error. The red highlighted expression is wrongly collected. The correct formulae should be $E_1 = D_0(H_2) - D_0(H_2^+) + I(H)$.

Structured Chemistry Reasoning with Large Language Models

Instruction	<p>Please provide a clear and step-by-step solution for a scientific problem in the categories of Chemistry, Physics, or Mathematics. The problem will specify the unit of measurement, which should not be included in the answer. Express the final answer as a decimal number with three digits after the decimal point. Conclude the answer by stating "The answer is therefore $\boxed{\text{ANSWER}}$."</p> <p>For each instance, you need to three things. Firstly, for "formulae retrieval", you need to identify the formulae explicitly and implicitly entailed in the problem context. Then there is a "reasoning/calculation process" where you are required to reason step by step based on the identified formulae and problem context. Finally, conclude the answer. For each problem, the output format should incorporate the following components in the corresponding format:</p>
Output Format	<pre> **Formulae retrieval:** [Formula 1] (the formula required to solve the problem) [Formula 2] (the second formula required to solve the problem, if any) ... [Formula n] (the n-th formula required to solve the problem, if any) **Reasoning/calculation process:** [step 1] (the first step for solving this problem) [step n] (the n-th step for solving the problem, if any) **Answer conclusion:** [answer] The answer is therefore $\boxed{\text{ANSWER}}$. </pre>
Demonstration (5 samples)	<p>To clearly explain the task, we provide the following example:</p> <p>Problem: Assume that all gases are perfect and that data refer to 298.15 K unless otherwise stated. Calculate the change in chemical potential of a perfect gas when its pressure is increased isothermally from 1.8 atm to 29.5 atm at 40°C. The unit of the answer is kJ mol^{-1}.</p> <p>Response: In order to solve this problem, we will use the formula for the change in chemical potential $\Delta \mu$ of a perfect gas due to a change in pressure. Given that the temperature is constant (isothermal), the chemical potential of a perfect gas is given by:</p> <pre> **Formula retrieval:** \[\text{[Formula 1]} \quad \Delta \mu = RT \ln\left(\frac{P_2}{P_1}\right)\] where: - $\Delta \mu$ is the change in chemical potential, - R is the universal gas constant ($8.314 \text{ J mol}^{-1} \text{ K}^{-1}$), - T is the absolute temperature in Kelvin, - P_1 is the initial pressure, - P_2 is the final pressure. **Reasoning/calculation process:** [step 1] Convert the temperature from Celsius to Kelvin. To convert from Celsius to Kelvin, we add 273.15 to the Celsius temperature. \[\text{[Equation 1]} \quad T = 40^\circ \text{C} + 273.15 \text{ K}\] \[\text{[Equation 2]} \quad T = 313.15 \text{ K}\] [step 2] Substitute the given values into the formula. \[\text{[Equation 3]} \quad P_1 = 1.8 \text{ atm}\] \[\text{[Equation 4]} \quad P_2 = 29.5 \text{ atm}\] \[\Delta \mu = 8.314 \text{ J mol}^{-1} \text{ K}^{-1} \cdot 313.15 \text{ K} \cdot \ln\left(\frac{29.5 \text{ atm}}{1.8 \text{ atm}}\right)\] [step 3] Calculate the natural logarithm and proceed with the calculation. \[\Delta \mu = 8.314 \text{ J mol}^{-1} \text{ K}^{-1} \cdot 313.15 \text{ K} \cdot \ln(16.389)\] \[\Delta \mu = 8.314 \text{ J mol}^{-1} \text{ K}^{-1} \cdot 313.15 \text{ K} \cdot 2.799\] \[\Delta \mu = 7293.269 \text{ J mol}^{-1}\] [step 4] Convert the result from Joules per mole to kilojoules per mole (1 kJ = 1000 J). \[\Delta \mu = 7.293 \text{ kJ mol}^{-1}\] **Answer conclusion:** \[\text{[answer]} \text{ The answer is therefore } \boxed{7.293 \text{ kJ mol}^{-1}}.\] </pre>
Trigger	<p>Following the above examples, please help me solve the following problem. Remember to strictly follow the output format.</p>

Figure 14. Full prompt used for generation.