AUTOPRM: AUTOMATING PROCEDURAL SUPERVISION FOR MULTI-STEP REASONING VIA CONTROLLABLE QUESTION DECOMPOSITION

Zhaorun Chen[*†1] **Zhuokai Zhao**[†1] **Zhihong Zhu**[†2] **Ruiqi Zhang**[3]
Xiang Li[4] **Bhiksha Raj**[4] **Huaxiu Yao**[5]
[1]University of Chicago, [2]Peking University [3]University of California, Berkeley
[4]Carnegie Mellon University [5]UNC-Chapel Hill
zhaorun@uchicago.edu, huaxiu@cs.unc.edu

ABSTRACT

Recent advancements in large language models (LLMs) have shown promise in multi-step reasoning tasks, yet their reliance on extensive manual labeling to provide procedural feedback remains a significant impediment. To address this challenge, in this paper, we propose a novel self-supervised framework **AutoPRM** that efficiently enhances the fine-tuning of LLMs for intricate reasoning challenges. Specifically, **AutoPRM** first decomposes complex problems into more manageable subquestions with a controllable granularity switch, then sequentially apply reinforcement learning to iteratively improve the subquestion solver. Additionally, we propose context-guided decoding to avoid reward tampering and guide the subquestion solver towards the solution of the holistic problem. Extensive experiments show that **AutoPRM** significantly improves performance on mathematical and commonsense reasoning tasks over SOTA. More encouragingly, **AutoPRM** can be easily integrated with other orthogonal reasoning pipelines.

1 INTRODUCTION

The landscape of natural language processing has been profoundly reshaped by the evolution of large language models (LLMs), which have demonstrated remarkable capabilities in a variety of complex tasks (Brown et al., 2020; Chen et al., 2021; Yuan et al., 2023). Among these, multi-step reasoning has emerged as a particularly challenging area and has drawn significant research attention (Bhattacharya, 2017; Hoffmann et al., 2022; Bubeck et al., 2023). To enhance the complex reasoning capabilities for LLMs, recent prompting-based approaches, including chain-of-thought (CoT) (Kojima et al.; Wei et al., 2022) and self-evaluation decoding (Wang et al., 2022; Yao et al., 2023; Xie et al., 2023), have proven to be successful. However, while being effective on large-sized models such as GPT-4 (OpenAI, 2023) and PaLM-2 (Anil et al., 2023), they are less effective for smaller, non-finetuned models such as GPT-3 (Brown et al., 2020) and LLaMA-2-7B (Touvron et al., 2023), which are poor reasoners by nature (Stolfo et al., 2022).

On the other hand, fine-tuning methods are also known to be effective for enhancing complex reasoning capabilities, especially for smaller-sized models (Uesato et al., 2022; Luo et al., 2023; Shridhar et al., 2023). Therein, procedural supervision-based fine-tuning (Wu et al., 2023; Lightman et al., 2023) has proved to be particularly effective (Uesato et al., 2022), which emulates human problem-solving process and provides step-by-step feedback, as opposed to outcome-based supervision which simply optimizes for the final-answers (Shridhar et al., 2023). Despite its advancements, the reliance on step-wise human annotations for these process-supervised reward models (PRM) presents a significant bottleneck. More specifically, such annotation is not only resource-intensive in terms of both time and domain expertise (Lightman et al., 2023), but also introduces human bias due to subjective judgements, which could potentially undermine the fine-tuning performance (Casper et al., 2023; Lightman et al., 2023).

[*]Work was done during Zhaorun Chen's remote internship at UNC-Chapel Hill.

[†]Lead authors.

To address these challenges, we propose a novel self-supervised procedural reward model termed **AutoPRM** to boost the efficacy in fine-tuning and inference for long-chained, complex reasoning problems. Concretely, AutoPRM first breaks down the problem into a sequence of sub-questions with a trained question-decomposition (QD) model, then solves each subquestion with a Reinforcement Learning (RL)-optimized question-answering (QA) model (Silver et al., 2017). More information on how AutoPRM builds upon yet differs from related works is discussed in Appendix A.

2 AUTOPRM

In this section, we detail our proposed framework **AutoPRM**. Our key insight is that automating step-wise question decomposition provides a natural perspective to reduce problem dimensions, through which model inference and optimization can be more precise and efficient. With decomposition, the fine-grained feedback can be obtained with a reliable step-wise verifier trained to predict intermediate results, which leads to a more powerful and bias-free reasoning model.

Preliminaries and Notations. Our problem formulation involves a dataset $\mathcal{D} = \{(p_i, a_i)\}_{i=1}^{N}$, where each problem p_i is associated with a final answer a_i that can be reached through reasoning. AutoPRM breaks the problem into multiple steps and models the reasoning process as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{Q}, \mathcal{R}, P, \gamma \rangle$ (Ramamurthy et al., 2022), where each MDP episode starts with a sampled problem input p_i and ends either when a final answer is generated or the model abstains. AutoPRM manages each individual reasoning step as a subquestion-solution pair $\{(q_t, s_t), q_t \in \mathcal{Q}, s_t \in \mathcal{S}\}$, where $\mathcal{S}$ is the state (subsolution) space and $\mathcal{Q}$ is the subquestion space.[1] The transition function appends a subsolution s_t to the end of the cumulative state $(p_i, s_0, s_1, \ldots, s_{t-1})$ at each step. A reward function (verifier) $\mathcal{R} : \mathcal{Q} \times \mathcal{S} \rightarrow \mathbb{R}$ can be either outcome-based (Cobbe et al., 2021), which provides a sparse feedback at the end of generation, or process-based (Uesato et al., 2022), which evaluates each step individually and assigns a fine-grained score for each intermediate step (q_t, s_t).

Collecting Subquestion-Subsolution Pairs. First, we illustrate the procedures to prepare the subquestion-subsolution dataset $\mathcal{D}_{\text{sub}} = \{\{q_{i,t}, s_{i,t}\}_{t=1}^{n_i}\}_{i=1}^{N}$. $\mathcal{D}_{\text{sub}}$ will be used to train the QD and QA models, which are the core parts of the proposed AutoPRM. Instead of manually providing subquestions as seen in prior works (Xie et al., 2023), AutoPRM adopts an efficient and unified framework to collect subquestions by training an auxiliary subquestion collection (SQC) Model. The core idea behind SQC is based on the assumption that each sentence in the groundtruth solution represents a valid step that progressively leads to the final solution (Lightman et al., 2023). In practice, we treat each sentence in the groundtruth solution as a subsolution, and take these subsolutions as inputs to SQC to generate the corresponding subquestions. To train this SQC model, we initially select a small subset of the original dataset $\mathcal{D}$ and prompt GPT-3.5 (prompts are detailed in Appendix D.1) to find appropriate subquestions that would reasonably induce these subsolutions. Next, we fine-tune an open-source Language Model (LM) (e.g. LLaMA-2 (Touvron et al., 2023)) on this training set to obtain the SQC model. We employ the SQC model to break each question-solution pair in the original dataset $\mathcal{D}$ into corresponding subquestions and subsolutions, yielding a new dataset $\mathcal{D}_{\text{sub}}$.

Reciprocal Question Answering. Fine-tuning LLMs via automatically generated intermediate solutions involves two parts: question decomposition (QD), which divides each question into a sequence of subquestions, and question answering (QA), which answers these subquestions and generates corresponding subsolutions. Both QD and QA naturally connect and influence each other. Therefore, to leverage the interconnections between them and better refine the overall effectiveness of AutoPRM, we propose *Reciprocal Question Answering (RQA)*, which is inspired and theoretically grounded by the cognitive learning theory (Xu et al., 2023). Different from SQC which relies on the ground truth solutions to obtain the subquestions, QD is designed to automatically break down any arbitrary question without access to the ground truths. Precisely, QD model takes each problem p_i as input and generates a set of decomposed subquestions $(q_t, s_t)_{t=1}^{n_i}$. As exclusively training this QD model may result in overfitting, we encompass the original QD objective with the additional QA objective, which aims to infer subsolutions based on the corresponding subquestions and their surrounding context. In practice, we adopt two separate prompting mechanisms specified for QD and QA (prompts detailed in Appendix D.2). For QD, during the training phase, we concatenate each subquestion with a special split token to form an ordered sequence of subquestions. And in inference, we use the same prompt but parse the output directly into a set of subquestions. In terms

[1] Note that while state $s_t \in \mathcal{S}$ should strictly refer to the cumulative subsolutions $\sum_0^t s_i$ (and action $a_i \in \mathcal{A}$ denotes the i^{th} subsolution), we represent s_t as the t^{th} subsolution for the simplification of notation.

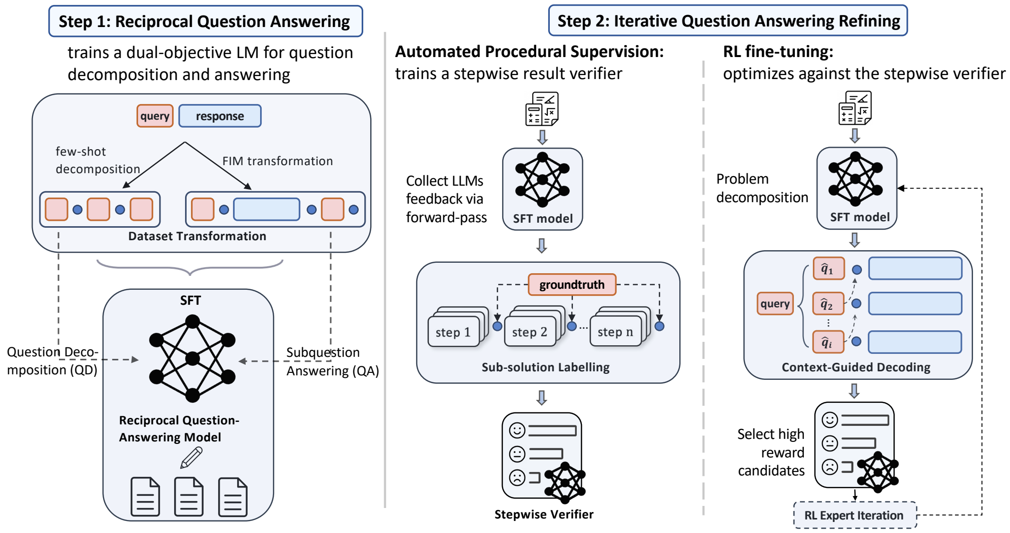
Step 1: Reciprocal Question Answering
trains a dual-objective LM for question decomposition and answering
query
response
few-shot decomposition
FIM transformation
Dataset Transformation
SFT
Question Decomposition (QD)
Subquestion Answering (QA)
Reciprocal Question-Answering Model
Step 2: Iterative Question Answering Refining
Automated Procedural Supervision:
trains a stepwise result verifier
Collect LLMs feedback via forward-pass
SFT model
groundtruth
step 1
step 2
step n
Sub-solution Labelling
Stepwise Verifier
RL fine-tuning:
optimizes against the stepwise verifier
Problem decomposition
SFT model
query
$\hat{q}_1$
$\hat{q}_2$
$\hat{q}_i$
Context-Guided Decoding
Select high reward candidates
RL Expert Iteration

Figure 1: Overview illustrating 3 steps of AutoPRM: (1) supervised fine-tuning (SFT) on $\mathcal{D}_{QG} \cup \mathcal{D}_{QA}$; (2) stepwise result verifier trained on the LLMs generated solutions of $\mathcal{D}_{QA}$; and (3) RL fine-tuning.

of QA, we propose *Context-Guided Decoding (CGD)* that is similar to the Fill-In-the-Middle (FIM) Decoding (Li et al., 2023; Liang et al., 2023), where each subquestion solver is guided by appending the subsequent subquestion to the beginning of the subsolutions that have being inferred. The detailed formulation of CGD is showed in Eq. (1)), with $\circ$ denoting the concatenation.

$$\langle \mathtt{PRE} \rangle \circ q_t \circ \langle \mathtt{SUF} \rangle \circ q_{t+1} \circ \langle \mathtt{MID} \rangle \circ [s_0, \ldots, s_{t-1}] \tag{1}$$

Through CGD, QA model is trained to derive holistically rational subsolutions that respond well to q_t while making good progress towards the final solution. We have the language modeling loss $\mathcal{L}(\mathcal{D}_{\text{sub}})$:

$$\mathcal{L}(\mathcal{D}_{\text{sub}}) = \sum_{i=1}^{N} \Bigg(\underbrace{-\sum_{t=1}^{n_i} \log P(q_t | q_{<t}, p_i)}_{\text{QD Loss}} \quad \underbrace{-\sum_{t=0}^{n_i} \log P(s_t | s_{<t}, q_t, q_{t+1})}_{\text{QA Loss}} \Bigg) \tag{2}$$

where $P(q_t|q_{<t}, p_i)$ is the probability of generating subquestion q_t conditioned on all the previous subquestions $q_{<t}$ and input problem p_i, aligning with the QD objective. And $P(s_t|s_{<t}, q_t, q_{t+1})$ is the probability of the model generating the subsolution s_t conditioned on all the preceding subsolutions $s_{<t}$, current subquestion q_t and subsequent subquestion q_{t+1}, aligning with the QA objective.

Empirically, we observe that some of the decomposed subquestion-subsolution pairs obtained in §2 are redundant and do not significantly contribute to the original final solutions. To this end, we propose a user-defined parameter $\epsilon \in [0, 1]$ to manage the granularity of the decomposition. Switching between decomposition granularity is tied to the specific choice of words within the problem context, which is equivalent to a linear transform in the embedding space (Han et al., 2023). We refer more details to Appendix D.4.

Automated procedural supervision. While the reciprocal QA model is sufficient to handle complex reasoning problems, they suffer from partial context and a loss of holistic view to obtain the final answer. Thus we aim to further enhance QA model with automated procedural supervision via RL. Specifically, this is achieved by first letting the QD model decompose problem p_i into intermediate subquestions, and then acquiring feedback r_t for each subsolution s_t via a step-wise binary verifier. As shown in Fig. 1, we implement a step-wise verifier (reward model, RM) as a language model to predict a binary label as either "correct" or "incorrect" after each step. To train this verifier, we first obtain a sample set of subquestion-subsolution pairs, as shown in the middle step of Fig. 1. Then the intermediate result of each subsolution is compared with the groundtruth and assigned values by:

$$I(\text{QA}(s_{t,i}), a_{t,i}) = \begin{cases} 1 & \text{if QA}(s_{t,i}) = a_{t,i}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

Algorithm 1 AutoPRM Decoding Procedure

```
Require: Fine-tuned model $\mathcal{M}$, problem $p$, granularity parameter $\epsilon$, abstaining threshold $\tau$.
The model $\mathcal{M}$ decomposes the problem $p$ into multiple sub-questions $\{q_1, q_2, ..., q_N\}$.
for $t = 1, ..., N$ do
    Take each one of the $k$ generated solution $s_{<t}$ as prefix, append $q_{t+1}$ to the end of $s_t$ via context-guided decoding
    Sample $m$ candidates of $s_t$
    if the score of all $mk$ candidates are less than $\tau$ then
        Abstain.
    else
        Choose top $k$ candidates of $s_t$ from all $mk$ samples according to the score Eq. (4).
    end if
end for
```

where $a_{t,i}$ is the intermediate groundtruth answer for the t^{th} subquestion of problem p_i. Notice that the policy that maximizes the score of intermediate steps also maximizes the RM-estimated probability of eventually reaching the correct final answer. Finally, the output label from Eq. (3) is appended after each subsolution and trained via the QA loss as defined in Eq. (2).

RL Fine-tuning. After obtaining the step-wise verifier, the last step of our pipeline, as showed in Fig. 1, is to apply RL via expert iteration (Silver et al., 2017) to further fine-tune the SFT models. Different from policy gradient methods, expert iteration alternates between policy improvement and policy distillation. In policy improvement, QA model produces k candidates for each problem p_i in the dataset $\mathcal{D}_{\text{sub}}$ via a decoding method. Then in policy distillation, we select the candidates with the highest scores based on the verifier and perform supervised-learning to improve the policy. When the training converges, we adopt Reward-Reranking (RR) decoding which selects the candidate subsolutions by RM-weighted probability (Uesato et al., 2022; Xie et al., 2023). Mathematically, we have:

$$s_t = \arg\max_{s_t} P_{\mathcal{M}}(s_t | s_{<t}, q_t, q_{t+1}) \cdot \mathcal{R}(s_t) \tag{4}$$

where $\mathcal{R}(s_t)$ is the probability for predicting "correct" by reward model $\mathcal{R}$. The insight here is that a correct reasoning step should be confirmed by both the inference model and the verifier. In the decoding process, RR is coupled with beam search (BS), a step-wise tree-based algorithm, to select the most probable sequence of words or tokens. Specifically in AutoPRM, as QA generates m candidate steps for each decomposed subquestion q, the top k candidates are selected according to their decoding scores s_t. This process is repeated until the model outputs the final answer or abstains from answering. The abstaining condition triggers when the scores of all mk candidates drop below the threshold τ at any step (Geifman & El-Yaniv, 2017). The complete AutoPRM decoding procedure for multi-step reasoning is illustrated in in Algorithm 1.

3 Experiments

Datasets. We assess AutoPRM on two arithmetic reasoning datasets, namely GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and one additional commonsense reasoning dataset, StrategyQA (Geva et al., 2021) in our experiments. We follow Shridhar et al. (2023) for preprocessing.

Baselines. We compare AutoPRM with a wide range of SOTA models. WizardMath (Luo et al., 2023) and MetaMath (Yu et al., 2023) are two SOTA models that enhance mathematical reasoning with external data augmentation. Distilling-LM (Shridhar et al., 2023) is a decomposition-based reasoning framework that adopts two separate models for QD and QA, and then distills (SFT) reasoning capabilities from GPT-3.5. We also compare reward-based (verifier-based) models such as ORM-RL (Cobbe et al., 2021) and PRM-RL (Lightman et al., 2023). Except for LLaMA-2-70B model that adopts a few-shot decoding without fine-tuning, all other approaches have been fine-tuned based on LLaMA-2-7B (Touvron et al., 2023)[2].

Experimental settings. To fairly compare with PRM baselines, we follow Uesato et al. (2022) and annotate a comparative amount of procedural feedback data equivalent to the subquestion-subsolution

[2] Since Distilling-LM did not publish its results on LLaMA-2 nor its training data, we report the result using our dataset.

Model	GSM8K	MATH
LLaMA-2 (70B)	56.8	13.5
LLaMA-2 (7B)	41.6	4.7
WizardMath*	54.9	10.7
MetaMath*	66.4	19.4
Distilling-LM	51.8	10.2
ORM-RL	52.9	6.9
PRM-RL	56.1	10.5
AutoPRM	**59.3 (+3.2%)**	**13.2 (+2.7%)**
AutoPRM*	**70.8 (+4.4%)**	**23.6 (+4.2%)**

Table 1: Comparison on GSM8K and MATH dataset. All models are fine-tuned based on LLaMA-2 7B by default. * indicates SFT on external data.

Approach	Final-Answer	Trace Result
SFT	56.8	61.0
SFT+ORM-RL	58.2	60.5
SFT+PRM-RL	65.1	66.0
SFT-AutoPRM-RL	**66.3 (+1.2)**	**67.4 (+1.4)**

Table 2: The accuracy of final-answer and stepwise (trace) result on StrategyQA dataset. The result indicates that while AutoPRM still improve upon ORM and PRM-based methods, its improvement is not as significant as in arithmetic reasoning tasks, probably due to model scale.

pairs used to train our step-wise verifier in AutoPRM. Detailed annotation procedures are shown in Appendix D.3. And the detailed hyperparameters of AutoPRM are reported in Appendix D.

Results. For arithmetic reasoning tasks, we have trained two sets of models: one is based on the groundtruth training dataset provided in GSM8K Cobbe et al. (2021) and MATH Hendrycks et al. (2021), and the other one is augmented using MetaMath Yu et al. (2023). The results are reported in Table 1. Table 1 indicates that AutoPRM achieves the best performance compared with other models. Specifically, AutoPRM reaches 59.3% on GSM8K and 13.2% on MATH with beam search, which outperforms PRM-RL by 3.2% and 2.7% respectively. Additionally, when applying our proposed pipeline to the MetaMath dataset for data augmentation and fine-tuning the model [3], AutoPRM's effectiveness is further improved, reaching 70.8% (+4.4%) on GSM8K and 23.6% (+4.2%) on MATH.

For the commonsense reasoning task tested with StrategyQA Geva et al. (2021), we consider the fine-tuned model on LLaMA-2-7B as our baseline. For ORM-RL, we directly train an outcome verifier based on the final binary prediction. And for PRM-RL, we follow Uesato et al. (2022) and annotate step-by-step to train a procedural supervised reward. The results are reported in Table 2. More detailed analysis is illustrated in Appendix B. These results highlight AutoPRM's substantial performance improvements and its versatility when combining with other methods.

Analysis. We further investigate the contributions of each sub-module of AutoPRM, the trace error as well as examine the correlation between AutoPRM's performance and problem length. The results are discussed in Appendix C.

4 CONCLUSIONS

In this paper, we introduce AutoPRM, a novel framework that automates procedural supervision for multi-step reasoning in LLMs. The core of AutoPRM consists of two reciprocal components: a QD model that systematically breaks down complex problems into manageable subquestions, and a QA model that accurately answers these subquestions. AutoPRM employs a robust training methodology, incorporating supervised fine-tuning, feedback-based step-wise verifier, and a final RL fine-tuning for the best performance. Through extensive experiments, we demonstrate that AutoPRM significantly outperforms SOTA methods in terms of efficiency and accuracy on three arithmetic and commonsense reasoning tasks. Results show that our automated QD process, coupled with a RL-optimized QA model, leads to a substantial improvement in handling complex reasoning tasks.

[3] https://huggingface.co/meta-math/MetaMath-7B-V1.0

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.

Arindam Bhattacharya. A survey of question answering for math and science problem. *arXiv preprint arXiv:1705.04530*, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Zhaorun Chen, Zhuokai Zhao, Tairan He, Binhao Chen, Xuhao Zhao, Liang Gong, and Chengliang Liu. Safe reinforcement learning via hierarchical adaptive chance-constraint safeguards, 2024a.

Zhaorun Chen, Zhuokai Zhao, Hongyin Luo, Huaxiu Yao, Bo Li, and Jiawei Zhou. Halc: Object hallucination reduction via adaptive focal-contrast decoding. *arXiv preprint arXiv:2403.00425*, 2024b.

Zhaorun Chen, Zhuokai Zhao, Wenjie Qu, Zichen Wen, Zhiguang Han, Zhihong Zhu, Jiaheng Zhang, and Huaxiu Yao. Pandora: Detailed llm jailbreaking via collaborated phishing agents with decomposed reasoning. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024c.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek Abdelzaher, and Heng Ji. Lm-switch: Lightweight language model conditioning in word embedding space. *arXiv preprint arXiv:2305.12798*, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*, 2022.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. A causal framework to quantify the robustness of mathematical reasoning with language models. *arXiv preprint arXiv:2210.12023*, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*, 2023.

Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. Badchain: Backdoor chain-of-thought prompting for large language models. *arXiv preprint arXiv:2401.12242*, 2024.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023.

Enwei Xu, Wei Wang, and Qingxia Wang. The effectiveness of collaborative problem solving in promoting students' critical thinking: A meta-analysis based on empirical literature. *Humanities and Social Sciences Communications*, 10(1):1–11, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. *arXiv preprint arXiv:2312.01823*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

A Related Work

LLMs struggle with complex reasoning tasks Lu et al. (2022). To mitigate this limitation, prompt-based methods including Chain-of-Thought (CoT) Wei et al. (2022) and its variants such as automatic CoT Zhang et al. (2022), Complex CoT Fu et al. (2022), Tree-of-Thought Yao et al. (2023), Graph-of-Thought Besta et al. (2023) and Exchange-of-Thought Yin et al. (2023) are developed. Although effective for larger LLMs, the performance of these methods is limited in smaller models OpenAI (2023); Anil et al. (2023); Lewkowycz et al. (2022), which leads to the exploration of problem decomposition into subquestions and sequential handling Wei et al. (2022); Gao et al. (2023); Chen et al. (2022; 2024c), coupled with step-wise feedback using self-verification Miao et al. (2023); Chen et al. (2024b;a), external LLMs Miao et al. (2023); Xie et al. (2023), heuristics Yao et al. (2023), and human-annotated rewards Uesato et al. (2022); Lightman et al. (2023). However, these methods often either depend on external large models Xiang et al. (2024), or require intensive human effort or specific designs Lightman et al. (2023), which severely limit their applicability.

Besides these prompt-based methods, fine-tuning has been another main line of research showing promise in enhancing LLMs reasoning capabilities for both large and smaller models Uesato et al. (2022); Luo et al. (2023); Shridhar et al. (2023), especially when pairing with data augmentation techniques such as multi-view question bootstrapping Luo et al. (2023) and instruction evaluation Yu et al. (2023). Among numerous fine-tuning approaches, existing studies suggest procedural supervision yields better accuracy than outcome-only methods Wu et al. (2023); Lightman et al. (2023); Shridhar et al. (2023). However, procedural supervision approaches often require extensive, unbiased manual labeling, which greatly limits their generalizability Uesato et al. (2022).

Inspired by the step-wise approaches as seen in the prompt-based methods, as well as procedural supervision used in fine-tuning approaches, in this paper, we introduce a novel self-supervised fine-tuning approach AutoPRM that significantly enhances LLMs reasoning capabilities while not requiring either external large models, or additional human efforts. Specially, inspired by human cognitive process where questioning and answering reciprocally enhances each other Xu et al. (2023), AutoPRM trains one unified model to handle both QD and QA.[4] Notably, instead of training a PRM that requires extensive manual annotations, AutoPRM adopts a more natural approach by directly training an intermediate outcome verifier to optimize the subquestion solver.

B Detailed Experimental Results

B.1 Results on Commonsense Reasoning

For the commonsense reasoning task tested with StrategyQA Geva et al. (2021), we consider the fine-tuned model on LLaMA-2-7B as our baseline. For ORM-RL, we directly train an outcome verifier based on the final binary prediction. And for PRM-RL, we follow Uesato et al. (2022) and annotate step-by-step to train a procedural supervised reward. The results are reported in Table 2.

The final-answer accuracy is verified with the groundtruth binary label. And to supervise step-wise (trace) result, we transform all open-ended subquestions to be close-ended, which can be answered with either "yes" or "no". Results confirm that procedural-supervised methods outperform outcome-based methods, and AutoPRM further achieves an accuracy gain of 1.2% on final-answer and 1.4% on intermediate result thanks to our precise and unbiased feedback. While the positive result indicates the effectiveness of our proposed method, the performance gain is less significant comparing to the large gains we observe in the arithmetic reasoning tasks.

We suspect that such result could be due to the model knowledge gap Petroni et al. (2019), as StrategyQA highly depends on factual truthfulness, which cannot be further enhanced by simply improving the reasoning framework. Therefore, we believe increasing model scale is the key factor to further improve accuracy on such knowledge-dependent reasoning tasks Anil et al. (2023).

C Additional Analysis on Experimental Results

In this subsection, we investigate the contributions of each sub-module to the overall improvement of AutoPRM, including decoding methods and controllable question decomposition. Additionally,

[4] Note that while we refer to QD and QA separately, they refer to the dual functions of a unified model in the remaining of the paper.

Approach	**GSM8K**	**MATH**
SFT	41.6	4.7
ORM-RL+Greedy	45.4	5.1
ORM-RL+SC	51.9	6.9
PRM-RL+Greedy	51.3	7.4
PRM-RL+BS	56.1	10.5
AutoPRM+Greedy	**52.8**	**10.4**
AutoPRM+BS	**58.2 (+2.1)**	**11.9 (+1.4)**
AutoPRM+RR	**58.9 (+2.8)**	**12.7 (+2.2)**
AutoPRM+GD	**59.3 (+3.2)**	**13.2 (+2.7)**

Table 3: Comparison of testing accuracy on two arithmetic dataset w.r.t. reward models and decoding strategies. All approaches is fine-tuned based on LLaMA-2-7B model and use naive greedy decoding by default. RR refers to Reward Ranking. (+Number) indicates the improvement performance compared to the best baseline PRM-RL+BS.

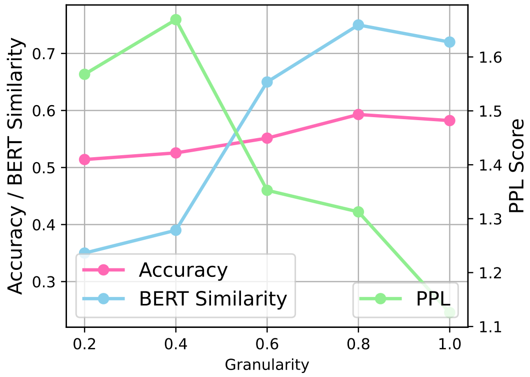

Figure 2: Assessment on GSM8K dataset w.r.t decomposition granularity ϵ. We evaluate the final-answer accuracy, perplexity and BERT similarity (to groundtruth solutions). Accuracy demonstrates that an intermediate granularity level (ϵ=0.8) yields best performance. Perplexity denotes that fine-grained guidance enhances the model's certainty in problem-solving. The increased similarity to the groundtruth solutions imply that AutoPRM effectively decompose questions that align with the human labeller.

we analyze the trace error as well as examine the correlation between AutoPRM's performance and problem length.

Decoding methods. First, we compare AutoPRM's performance against various reward models and decoding strategies. To start, we analyze the performance of ORM, PRM and AutoPRM coupled with different decoding strategies. From the results in Table 3, we can observe that: Since ORM-RL is an outcome-supervised method, we apply self-consistency and use the majority answer as the final-answer. All other methods apply a beam search process to yield the final-answer. The beam search baseline selects candidates by maximizing the RM score at each step, while the Reward-Reranking (RR) decoding, as outlined in Xie et al. (2023), maximizes the RM-reweighted score in Eq. (4).

Notice that RR achieves a better performance than beam search since they can select a more rational candidate by incorporating signals from both the LM and verifier. Similar to the RR decoding, our CGD decoding, which guides the individual subsolutions towards solving the original problem, can further mitigate the intrinsic issues brought by decomposition (*e.g.*, diminished context, topic deviation) and effectively enchance final performance.

Decomposition granularity. We evaluate AutoPRM performance across varying decomposition granularity on GSM8K. As shown in Fig. 2, while we can see that more precise and fine-grained decomposition generally leads to better accuracy and more certain factual inference Chuang et al. (2023), interestingly, an intermediate level of granularity ($\epsilon = 0.8$) achieves the highest accuracy in final answers. Additionally, the increased similarity to the groundtruth solutions indicate that our model can effectively break down questions in a manner that aligns with human cognitive process to solve multi-step reasoning problems.

Approach	**GSM8K**	**MATH**
SFT+ORM-RL	53.4	6.5
SFT+PRM-RL	70.5	13.1
AutoPRM-SFT	71.2	14.5
AutoPRM-SFT+RL	**72.0**	**14.7**

Table 4: Comparison of different fine-tuning methods in solving the decomposed dataset of subquestion-solution pairs. The results indicate that models with process-supervision can also solve the individual subquestions effectively. Conversely, the poor performance of ORM methods in solving the subquestions confirms that direct fine-tuning against ORM can lead to correct final-answer with the incorrect reasoning trace Uesato et al. (2022).

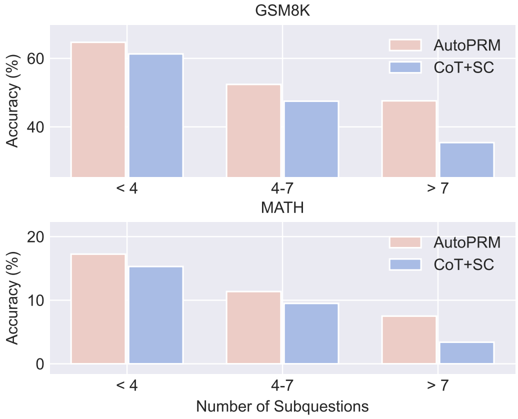

Figure 3: Comparing AutoPRM and CoT+SC decoding on problems of varing complexity and with different number of subquestions. AutoPRM outperforms CoT+SC by a large margin, especially for problems with longer reasoning chains.

Step-wise (Trace) Error Analysis. To evaluate the internal reasoning reliability, we assess the performance on each decomposed subquestion-subsolution pair and present the results in Table 4. Results show that while process-supervised models adeptly solves subquestions, AutoPRM gains more from precise, unbiased, and fine-grained feedback. Additionally, the fact that ORM-based methods only show slight improvements in original question solving (Table 3), suggests that they struggle to link correct final answers with their reasoning traces, which is also discussed in Lightman et al. (2023).

Varying problem length. In this section, we access the reasoning performance with respect to the length of the questions, which is a natural reflection of the question complexity as well as an approximation of the amount of reasoning needed to derive the final answers.

The results are showed in Fig. 3. In general, we notice that the performance gain (the absolute accuracy gain over self-consistency) increases as the reasoning chain becomes longer, which verifies the effectiveness of our method in guiding the reasoning trace to attain the correct final-answer, especially for longer-chained problems.

D Detailed Training Procedures

The complete procedures and hyperparameters (Table 5) for fine-tuning AutoPRM are detailed in this section. All models including SFT and RL-fine-tuned models are fully fine-tuned based on LLaMA-2-7B Touvron et al. (2023). Specifically, the RL via expert iteration process is iterated five epochs, with the best model being selected based on its performance in final-answer error on the validation set. All model training was conducted using Huggingface Library Wolf et al. (2020).

Parameters	**Value**
learning rate (SFT)	1e-4
learning rate (RL)	5e-5
learning rate scheduler	cosine
batch size	32
weight decay	0.05
warmup steps	100

Table 5: AutoPRM hyperparameter settings

D.1 DATA RETRIEVAL PROMPTS

D.1.1 SUBQUESTION COLLECTION (SQC) PROMPTS

Here is is a subsolution to a grade school math question. You should first (1) rephrase information (e.g. numbers, conditions) from the context necessary to reconstruct the subsolution, (2) delete redundant information not used in the subsolution, (3) ask a subquestion based on this subsolution. Please make sure you include all the necessary information in the subsolution.

Example 1:

Context: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg.

Subsolution: Janet sells 16 - 3 - 4 = «16-3-4=9»9 duck eggs a day.
Subquestion: Janet's ducks lay 16 eggs per day. She eats three herself and bakes muffins with four. She sells the remainder to the market. How many duck eggs does Janet sell a day?

Example 2:

Context: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed.

Subsolution: If each chicken eats 3 cups of feed per day, then for 20 chickens they would need 3*20=«3*20=60»60 cups of feed per day.
Subquestion: Each day Wendi feeds each of her chickens three cups of mixed chicken feed. How many cups of feed do 20 chickens need per day?
Now let's find the subquestion for some subsolutions!
Context:
Subsolution:
Subquestion:

Table 6: The prompt input to GPT-3.5 for subq-question collection on GSM8K dataset

Here is a solution of multiple steps to a grade school math question. Please break the question down into several intermediate questions that ask the result of each intermediate step of the solution. You should provide the context of the original question first, then provide the intermediate questions and corresponding interemediate solutions. For example:

Original Question: Lana is brewing cups of tea for her friends. She has 27 cups, and she divides these into 3 rows. In each row, she creates equal amounts of chamomile and mint tea cups. She then uses the remaining cups to brew a total of 15 cups of cinnamon tea. How many cups of mint tea are in each row?

Original Solution: If there are 15 cups of cinnamon tea, then there are a total of 27 - 15 = «27-15=12» 12 cups of chamomile or mint tea.
As there are equal amounts of chamomile tea and mint tea, there is a total of 12 cups / 2 = «12/2=6»6 cups of mint tea.
Dividing these into rows shows that each row holds 6 / 3 = «6/3=2»2 cups of mint tea.
2
The answer is: 2

Break Down:
Context: Lana is brewing cups of tea for her friends. She has 27 cups, and she divides these into 3 rows. In each row, she creates equal amounts of chamomile and mint tea cups. She then uses the remaining cups to brew a total of 15 cups of cinnamon tea.

Intermediate Question 1: How many cups of chamomile or mint tea are there?
Intermediate Solution 1: If there are 15 cups of cinnamon tea, then there are a total of 27 - 15 = «27-15=12»12 cups of chamomile or mint tea.
12
The answer is: 12

Intermediate Question 2: How many cups of mint tea are there?
Intermediate Solution 2: If there are 15 cups of cinnamon tea, then there are a total of 27 - 15 = «27-15=12»12 cups of chamomile or mint tea.
As there are equal amounts of chamomile tea and mint tea, there is a total of 12 cups / 2 = «12/2=6»6 cups of mint tea.
6
The answer is: 6

Intermediate Question 3: How many cups of mint tea are in each row?
Intermediate Solution 3:If there are 15 cups of cinnamon tea, then there are a total of 27 - 15 = «27-15=12»12 cups of chamomile or mint tea.
As there are equal amounts of chamomile tea and mint tea, there is a total of 12 cups / 2 = «12/2=6»6 cups of mint tea.
Dividing these into rows shows that each row holds 6 / 3 = «6/3=2»2 cups of mint tea.
2
The answer is: 2

Please strictly follow the format I give you.

Table 7: The prompt input to GPT-3.5 for question decomposition on GSM8K dataset

Let's generate the subquestions(or sub-instructions) and subsolutions to obtain the final answer for this math problem. Use exactly one operation per step. Mathematical expression should be in latex format (e.g. $\binom{5}{2}5^3$). Put your final answer in a box (e.g. $\boxed{\frac{625}{648}}$).

Example 1:

Original Question: Find the value of n that satisfies $2(n+1)!+6n! = 3(n+1)!$, where $n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$.
Groundtruth answer: 5
Subquestion 1: Move all terms to the right side.
Subsolution 1: Moving all terms to the right side:

$$0 = 3(n+1)! - 2(n+1)! - 6n!$$

$$\boxed{0 = (n+1)! - 6n!}$$

Subquestion 2: Take out a factor of $n!$.
Subsolution 2:

$$0 = n!(n+1-6)$$
$$0 = n!(n-5)$$

Subquestion 3: Divide out $n!$.
Subsolution 3: We know that $n! \neq 0$, so we can divide out $n!$ and solve for n:

$$0 = n-5$$
$$n = \boxed{5}$$

Example 2:

Original Question: Steve has one quarter, two nickels and three pennies. Assuming no items are free, for how many different-priced items could Steve individually pay for with exact change?
Groundtruth answer: 23
Subquestion 1: How many possibilities does the quaters contribute?
Subsolution 1: Steve can use no quarters or one quarter, for $\boxed{2}$ possibilities.
Subquestion 2: How many possibilities do the nickels provide?
Subsolution 2: Steve can use 0, 1, or 2 nickels, for $\boxed{3}$ possibilities.

Subquestion 3: How many possibilities will the pennies provide?
Subsolution 3: Steve can use 0, 1, 2, or 3 pennies, for $\boxed{4}$ possibilities.
Subquestion 4: How many possibilies in total?
Subsolution 4: That gives $2 \cdot 3 \cdot 4 = 24$ possible combinations. But we must remove the combination where Steve does not use any coins, leaving us with $24 - 1 = \boxed{23}$.

Original Question:
Groundtruth answer:

Table 8: The prompt input to GPT-3.5 for question decomposition on MATH dataset

Here are several facts that will help answer a question. Please organize an inference with the given facts to answer the question, with an explicit answer of either True or False at the end. Then break the question down into several intermediate questions that ask the stage of each intermediate step of your inference. For example:

Original Question: Do the anchors on Rede Globo speak Chinese?

Facts:
1. Rede Globo is a Brazilian television network.
2. The official language of Brazil is Portuguese.

Inference Solution: No. Rede Globo is a Brazilian television network, and Brazil's official language is Portuguese. Thus anchors on Rede Globo do not speak Chinese. The answer is: False.

Break Down:
Intermediate Question 1: What country broadcasts Rede Globo?
Intermediate Solution 1: Rede Globo is a Brazilian television network.
The answer is: Brazil.

Intermediate Question 2: What is the official language of Brazil?
Intermediate Solution 2: The official language of Brazil is Portuguese.
The answer is: Portuguese.

Intermediate Question 3: Is Portuguese Chinese?
Intermediate Solution 3: The Portuguese is not Chinese.
The answer is: False.

Please strictly follow the format I give you. Generate the Inference Solution first, and then break down the question into several Intermediate Question and Intermediate Solution.

Table 9: The prompt input to GPT-3.5 for question decomposition on StrategyQA dataset

D.2 AutoPRM Model Prompt

D.2.1 Question Decomposition (QD) Model Prompt

Let's break down this question into a chain of subquestions.
Context: John is buying a new pair of shoes that costs $95. He has been saving up his money each month for the past three months. He gets a $5 allowance a month. He also mows lawns and shovels driveways. He charges $15 to mow a lawn and $7 to shovel. After buying the shoes, he has $15 in change.
Question: If he mows 4 lawns, how many driveways did he shovel?

Chain of subquestions: How much money did John save up in total? -> How much money did John save from his allowance? -> How much money did John earn from mowing lawns? -> How much money did John earn from shoveling driveways? -> How many driveways did he shovel?

Table 10: The prompt input to AutoPRM for question decomposition (QD)

D.2.2 Question Answering (QA) Model Prompt

Below is a math question. Write a solution that answers to the question. The solution may not use all the conditions provided in the question.
Question:
Solution

Table 11: The prompt input to AutoPRM for question answering (QA)

D.3 Process-supervised Data Annotations

In this section, we detail the data annotation procedure to train Process-supervised Reward Model (PRM) baselines Lightman et al. (2023); Uesato et al. (2022) to compare with our proposed model. As outlined in §3, the PRM is trained using step-wise labels to assess the correctness of each step. We gather this data by having human annotators review the original question and standard solution from the arithmetical and commonsense reasoning dataset (GSM8K Cobbe et al. (2021), MATH Hendrycks et al. (2021), and StrategyQA Geva et al. (2021)), as well as the solution generated by the model. Specifically, annotators are asked to identify the first step in the model solution that contains a significant error, if any. A significant error, as defined in existing works Uesato et al. (2022), is a step where the reasoning is either incorrect or makes it impossible to reach the correct solution anymore without revising that step. Based on these assessments, each step receives a binary label: steps preceding the first significant error are marked as 'correct', while subsequent steps are labeled 'incorrect'.

D.4 Data Preparation for Interpreting Decomposition Granularity

When $\epsilon = 1$, the original question-solution pairs are fully-decomposed into the subquestion-subsolution pairs, while decomposition with $\epsilon = 0$ being practically the same as the one-shot CoT results. For an intermediate ϵ, we select a subset of $\tilde{n}_i$ subquestions from the fully-decomposed pairs that significantly contribute to the final answer via a heuristic, and assign $\epsilon = \tilde{n}_i / n_i$. Finally, this ϵ is integrated into the QD prompt and optimized using Eq. (2).

To prepare data to train AutoPRM to interprete with the decomposition granularity parameter, we follow Han et al. (2023) and assign $\epsilon = 1$ to all the fully-decomposed subquestion-subsolution pairs and $\epsilon = 0$ to the one-shot CoT prompt of each problem p_i. Specifically, for an intermediate ϵ, we select a subset of $\tilde{n}_i$ subquestions from the fully-decomposed pairs that significantly contribute to the final answer via a heuristic, and assign $\epsilon = \tilde{n}_i / n_i$.

The heuristic determines if a subsolution in a reasoning process is important by checking if it introduces a new condition or calculation into the reference context. Specifically, we adopt a simple token-matching method via regex to sequentially check for new conditions. For example, for GSM8K we extract two types of tokens: entities and numbers. Sequentially, we check if each subsolution in the decomposed subquestion-solution pairs set introduces a new entity (implying new condition) or new number (implying calculation). If yes, we consider this subsolution as contributive to the final-answer. Finally, this ϵ is integrated into the QD prompt and optimized using Eq. (2).