SEER: Facilitating Structured Reasoning and Explanation via Reinforcement Learning

Guoxin Chen[†§], **Kexin Tang**[*], **Chao Yang**[†✉], **Fuying Ye, Yu Qiao**[†], **Yiming Qian**[‡✉]
[†]Shanghai Artificial Intelligence Laboratory
[§]Institute of Computing Technology, Chinese Academy of Sciences
[*]Shanghai Jiao Tong University
[‡]Agency for Science, Technology and Research (A*STAR)
chenguoxin22s@ict.ac.cn, {tkx94china, fuyingye.work}@gmail.com,
{yangchao,qiaoyu}@pjlab.org.cn, qiany@ihpc.a-star.edu.sg

Abstract

Elucidating the reasoning process with structured explanations from question to answer is crucial, as it significantly enhances the interpretability, traceability, and trustworthiness of question-answering (QA) systems. However, structured explanations demand models to perform intricately structured reasoning, which poses great challenges. Most existing methods focus on single-step reasoning through supervised learning, ignoring logical dependencies between steps. Moreover, existing reinforcement learning (RL) based methods overlook the structured relationships, underutilizing the potential of RL in structured reasoning. In this paper, we propose SEER, a novel method that maximizes a structure-based return to facilitate structured reasoning and explanation. Our proposed structure-based return precisely describes the hierarchical and branching structure inherent in structured reasoning, effectively capturing the intricate relationships between different reasoning steps. In addition, we introduce a fine-grained reward function to meticulously delineate diverse reasoning steps. Extensive experiments show that SEER significantly outperforms state-of-the-art methods, achieving an absolute improvement of 6.9% over RL-based methods on EntailmentBank, a 4.4% average improvement on STREET benchmark, and exhibiting outstanding efficiency and cross-dataset generalization performance. Our code is available at https://github.com/Chen-GX/SEER.

1 Introduction

Navigating machines to understand and articulate the thought process from posing a question to arriving at an answer has been a long-term pursuit in the AI community (McCarthy, 1959; Yu et al., 2023). Current QA explainable systems adeptly furnish brief supporting evidence (Rajani et al., 2019; DeYoung et al., 2020). However, they often fail to clarify the *reasoning process* from prior knowledge to the derived answer. By elucidating the reasoning process of answers generation from the language models, we can greatly improve interpretability, trustworthiness, and debuggability (Dalvi et al., 2021; Ribeiro et al., 2023). As illustrated in Figure 1, when generating answers for the question "Which natural material is best for making a table?", the reasoning process with structured explanations, such as entailment trees (Dalvi et al., 2021) or reasoning graphs (Ribeiro et al., 2023), explains why "sturdy wood" is the best answer.

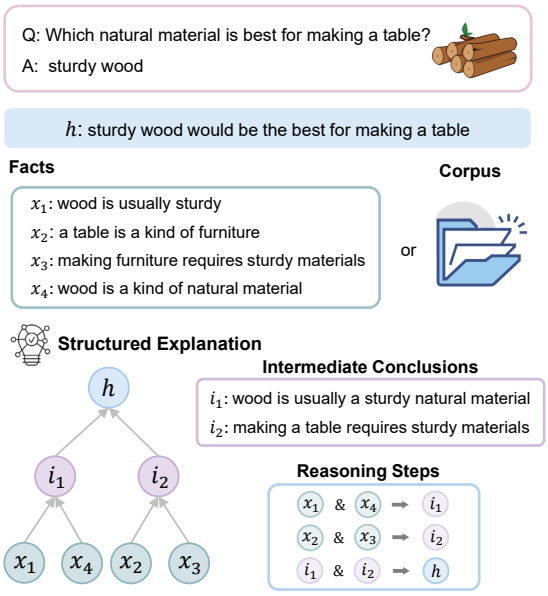


Figure 1: An example of structured explanation. Given a hypothesis h (a declarative sentence derived from a question-answer pair) and a set of facts (or corpus), the goal is to generate a structured explanation, which delineates the reasoning process from facts to the hypothesis.

Deriving such complex structured explanations poses a great challenge. Previous methods (Dalvi et al., 2021; Tafjord et al., 2021) consider structured explanations as linearized sequences and generate the entire reasoning process in one go. However, these methods lack controllability and may hallucinate unreliable reasoning steps. To ad-

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5901–5921
August 11-16, 2024

Method	Training Emphasis	Runtime	Return
RLET	multi-step reasoning	9.34s	chained
FAME	single-step reasoning	30.77s	/
Ours	structured reasoning	3.91s	structured

Table 1: Comparative analysis of different methods: RL-based method, RLET (Liu et al., 2022), supervised method, FAME (Hong et al., 2023), and our approach.

dress these concerns, recent studies (Hong et al., 2022; Neves Ribeiro et al., 2022; Yang et al., 2022) decompose structured explanations and focus on single-step reasoning via supervised learning. Nevertheless, this kind of approach may not always yield optimal results as they fail to consider the interdependencies between different steps. FAME (Hong et al., 2023) attempts to compensate for these shortcomings by leveraging Monte-Carlo planning (Kocsis and Szepesvári, 2006), which significantly increases the running time and inadvertently explores numerous ineffective steps (as shown in Table 1). Furthermore, FAME still concentrates on isolated single-step reasoning, which lacks support for structured reasoning. As a general framework for solving sequential decision-making problems, reinforcement learning (RL) is employed in RLET (Liu et al., 2022) to enhance multi-step reasoning. However, RLET defines the return (a.k.a. cumulative reward) using the standard chain structure, thus lacking the ability to represent the tree (Dalvi et al., 2021) or graph (Ribeiro et al., 2023) logical structures inherent in structured reasoning. As a result, the potential of RL for structured reasoning is not fully exploited.

To address the above issues, we propose SEER, a novel method that *facilitates **S**tructured r**E**asoning and **E**xplanation via **R**einforcement learning*. In structured reasoning, we observe that the logical dependencies between different steps no longer follow a chained trajectory but instead adhere to the inherent tree or graph structure. Therefore, we propose the structure-based return to precisely describe a tree or graph logical structure, effectively capturing the complex interdependencies between different steps. Additionally, we refine the reward function to meticulously delineate diverse reasoning steps, specifically targeting redundant ones that do not contribute to the final structured explanations. Through experiments in Sec. 5.4, we find that redundant steps represent the exploration in the environment, and appropriate penalization contributes to improved reasoning performance.

Our contributions are summarized as follows:

- We propose SEER, a novel RL-based method that facilitates structured reasoning and explanation. To our knowledge, SEER is the first general framework that accommodates scenarios of chained, tree-based, and graph-based structured reasoning.
- We propose the structure-based return to address the intricate interdependencies among different reasoning steps, effectively stimulating the potential of RL in structured reasoning.
- We conduct extensive experiments to demonstrate the superiority of SEER over state-of-the-art methods. Our method facilitates the effectiveness and efficiency of structured reasoning and exhibits outstanding cross-dataset generalization performance.

2 Related Work

2.1 Explanation for Question Answering

Extensive research has delved into various forms of interpretability in QA systems (Thayaparan et al., 2020; Wiegreffe and Marasovic, 2021; Lamm et al., 2021; Chen et al., 2023). Different from the free-form texts susceptible to hallucinations (Rajani et al., 2019; Wei et al., 2022) or the rationales that only provide supporting evidence (DeYoung et al., 2020; Valentino et al., 2021), the structured explanations, such as the entailment trees (Dalvi et al., 2021) and reasoning graphs (Ribeiro et al., 2023), offer a novel way to generate explanations. These structured methods utilize tree or graph formats to clearly outline *what* information is used and *how* it is combined to reach the answer. Despite the remarkable interpretability, the intricately structured reasoning also poses significant challenges (Yu et al., 2023; Xu et al., 2023).

2.2 Natural Language Reasoning

Natural language reasoning, a process that integrates multiple knowledge to derive new conclusions, has attracted significant attention (Saha et al., 2020; Tafjord et al., 2021; Sanyal et al., 2022; Chen et al., 2024). Among these, the entailment trees and reasoning graphs, which involve structured reasoning and reasoning path generation tasks, present considerable challenges (Yu et al., 2023). Dalvi et al. (2021) attempt to transform structured reasoning into a linearized sequence to fit generative models, which may generate hallucinations and invalid reasoning. To alleviate this issue, recent studies (Neves Ribeiro et al., 2022; Hong et al., 2022; Neves Ribeiro et al., 2022; Hong et al., 2023) per-

form premises selection and reasoning in a step-by-step manner. Nevertheless, these methods decompose structured reasoning and solely leverage isolated single-step supervision to train models. This kind of approach neglects the interdependencies between different steps, which may not always yield optimal results. Therefore, in light of the advancements of RL in various reasoning tasks (Poesia et al., 2021; Le et al., 2022), RLET (Liu et al., 2022) attempts to incorporate RL into the entailment trees. However, it has to enumerate all potential actions, which is unacceptable for practical scenarios. Furthermore, RLET still defines returns in chained trajectories to facilitate multi-step reasoning, which is not suitable for tree/graph-based structured reasoning. In contrast, our SEER showcases superior adaptability to chained, tree-based, and graph-based structured reasoning via the structure-based return, which significantly enhances both the reasoning performance and efficiency.

3 Method

3.1 Task Formulation

As illustrated in Figure 1, the input of the task comprises a set of facts $X = \{x_1, x_2, \ldots, x_n\}$ and a hypothesis h. The output of the task is the reasoning steps in a structured form, such as an entailment tree T or a reasoning graph[1]. The entailment tree T consists of tree-structured reasoning, whose leaf nodes are selected from the relevant facts (x_*) and intermediate nodes represent the derived intermediate conclusions (i_*). We represent the annotated ground-truth entailment tree as T_{gold}, with its leaf nodes signifying X_{gold}.

3.2 Overview

We model the structured reasoning as a reinforcement learning (RL) task, the goal of which is to learn the optimal reasoning policy. Figure 2 illustrates the overall framework of SEER, which mainly includes trajectory rollout and policy optimization. For trajectory rollout, we generate trajectories based on the current policy, and each trajectory is produced iteratively until the stopping criteria are satisfied (Appendix C.1). For policy optimization, we assign rewards to the collected trajectories and update both the policy and critic using the structure-based return. Algorithm 1 (Appendix A) outlines our proposed method for further reference.

[1]Although the reasoning graph (Ribeiro et al., 2023) is a more general structure, to be consistent with the majority of previous work, we use the entailment tree (Dalvi et al., 2021) as an example to formalize the task and illustrate our method. Our proposed method is also applicable to the task described in the form of a reasoning graph.

3.3 Fine-grained Component of SEER

State At reasoning step t, we define the state $s_t = \{h, P_t, C_t\}$ as a combination of the hypothesis h, existing reasoning steps P_t and candidate sentences C_t. P_t contains the reasoning steps so far, and C_t is the set of sentences that can be selected as premises. Each sentence in C_t is either unused facts or intermediate conclusions I_t generated by previous steps, i.e., $C_t = \{X \cup I_t \backslash U_t\}$, where U_t is the set of used sentences. For the initial state, $s_1 = \{h, P_1 = \varnothing, C_1 = X\}$.

Action Given the state s_t, we consider two types of actions $a_t \in \mathcal{A}(s_t)$: (1) "`Reason: <premises>`": the entailment module is invoked to generate a new intermediate conclusion i_t based on the given <premises>. Here, <premises> are selected from C_t. Then, the state is updated as follows: $P_{t+1} = P_t \cup \{\text{<premises>} \rightarrow i_t\}$, $U_{t+1} = U_t \cup \{\text{<premises>}\}$, and $I_{t+1} = I_t \cup \{i_t\}$. (2) "`End`": This action signifies the end of the reasoning process and returns the trajectory τ.

Policy The action type "`Reason: <premises>`" induces a large action space, since premises can be any combination of sentences from the candidate set C. To enumerate the probabilities of all potential actions and then sample an action to execute, previous studies (Liu et al., 2022; Hong et al., 2022) limit combinations to pairwise premises, such that the action space is reduced to $\binom{n}{2}$, where n is the size of the set C. However, such a simplification incurs some potential drawbacks. First, as the number of candidate sentences increases, the number of potential actions grows exponentially. This renders them impractical for complex reasoning tasks with limited computational resources. Second, by restricting combinations to pairs only, the interdependencies among multiple premises are ignored, which may limit the effectiveness and richness of the derived conclusions.

To address this issue, we adopt a generative model to represent the policy π, which can directly sample from the action space $\mathcal{A}(s_t)$. Using the generative model essentially expands the action space where the combinations of premises can be arbitrary. This enables the policy to extensively

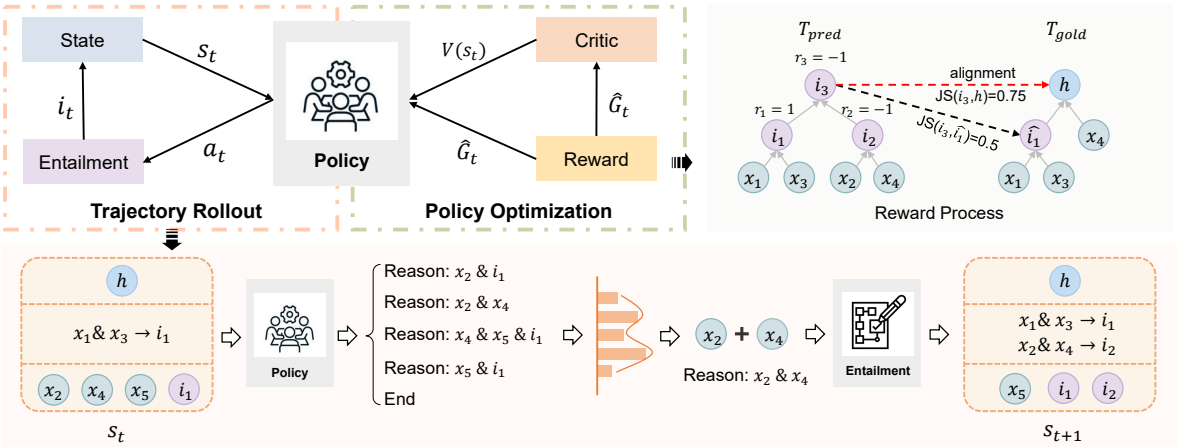


Figure 2: Overall framework of SEER. For trajectory rollout, action generation (Policy) and conclusion generation (entitlement) are performed alternately. The orange area details the reasoning process from s_t to s_{t+1}. For policy optimization, the reward module assigns rewards and updates the policy and critic based on tree or graph structures.

explore better actions during RL training, not limited to paired premises. Further, to speed up RL training, we first generate the top-k actions using policy π:

$$a_t^1, a_t^2, ..., a_t^k \sim \pi(a|s_t), \quad a \in \mathcal{A}(s_t), \quad (1)$$

where the input is a linearized state s_t (i.e., the concatenation of h, P_t, and C_t). Then, we proceed with re-normalization to form an appropriate probability distribution over the top-k actions, and sample from it to select the action a_t to be performed in the current reasoning step, that is,

$$\pi'(a_t^i|s_t) = \frac{\pi(a_t^i|s_t)}{\sum_{j=1}^{k} \pi(a_t^j|s_t)}, \quad i = 1, ..., k, \quad (2)$$

$$a_t \sim \pi'(a|s_t), \quad a \in \{a_t^1, a_t^2, ..., a_t^k\}. \quad (3)$$

Entailment Module If the action a_t is "Reason: <premises>", we invoke the entailment module to derive the intermediate conclusion to obtain the next state. The entailment module is also a generative model with its input being <premises>. Following Hong et al. (2022); Liu et al. (2022), we fine-tune the entailment model in a supervised manner and freeze the parameters during the reinforcement learning process, as shown in Figure 2.

Reward To evaluate the correctness of the entailment tree, Dalvi et al. (2021) proposed an alignment algorithm based on Jaccard similarity to align each intermediate node of the predicted tree T_{pred} with T_{gold}. However, different from the fully supervised learning methods, we observe that during the RL process, the policy explores different actions to identify the optimal reasoning process, inevitably attempting some redundant steps that do not contribute to reaching the final hypothesis. Existing RL-based work (Liu et al., 2022) simply treats redundant steps with the same penalty as erroneous steps. This simplification may negatively affect the learning process which discourages necessary exploration in the action space. Furthermore, it lacks detailed feedback to guide the policy toward optimal policy, as it fails to differentiate between innocuous actions (redundant steps) and incorrect actions (erroneous steps).

To this end, we propose a fine-grained reward function that assigns different reward values for correct steps, erroneous steps, and redundant steps, as shown in Equation 4. For a trajectory τ, we assume that the last intermediate conclusion is our predicted hypothesis since the policy deems it should End here. Then, we backtrack to construct the predicted entailment tree T_{pred} (see Appendix C.6 for more details). Note that there might be some steps not participating in T_{pred}, which are regarded as redundant steps. Then, as illustrated in Figure 5, we consider steps that perfectly match via the alignment algorithm (Dalvi et al., 2021) as correct steps and regard others as erroneous steps.

$$r_t = \begin{cases} 1, & \text{if perfectly match,} \\ -0.5, & \text{if } i_t \notin T_{\text{pred}}, \\ -1, & \text{otherwise.} \end{cases} \quad (4)$$

Critic To enhance training stability, we introduce the critic to estimate the state-value function $V(s_t)$. The input of $V(s_t)$ is a linearized state, and its

output is a scalar representing the return (i.e., cumulative reward) when starting from state s_t. In the simplest case, the return is the chained sum of the rewards. Accordingly, one-step temporal difference (TD) (Sutton, 1988) is often used to estimate $V(s_t)$, which is updated by the TD-target:

$$G_t = r_t + \gamma V(s_{t+1}), \tag{5}$$

where γ is the discount factor. However, in structured reasoning, reasoning steps typically adhere to inherent tree (Dalvi et al., 2021) or graph (Ribeiro et al., 2023) structures, with the chained structure being merely a special case. Thus, Equation 5 just describes the chained multi-step reasoning, which may not effectively capture the intricate logical dependencies between steps in structured reasoning.

Therefore, we propose the structure-based return, where the TD-target is expressed in a more general formulation:

$$\hat{G}_t = r_t + \gamma \frac{1}{|\mathcal{P}(s_t)|} \sum_{s_j \in \mathcal{P}(s_t)} V(s_j), \tag{6}$$

where $\mathcal{P}(s_t)$ represents the parent node of state s_t in the entailment tree T_{pred} or reasoning graph. When $s_t \notin T_{\text{pred}}$, $\mathcal{P}(s_t) = s_{t+1}$. It can be seen that our structure-based return (Equation 6) adapts to structured reasoning involving chained, tree-based and graph-based structured scenarios. Especially, entailment tree is a special case of the reasoning graph, in which each state typically has only one parent node, and thus Equation 6 degenerates into $\hat{G}_t = r_t + \gamma V(\mathcal{P}(s_t))$. Furthermore, as shown in Figure 6 (Appendix E), for equivalent trajectories $s_1 \to s_2 \to s_3 \to s_4$ and $s_2 \to s_1 \to s_3 \to s_4$, previous method (Liu et al., 2022) would assign different returns for state s_1 and s_2, even though they represent the same tree in the end. Conversely, our method, by precisely delineating the intricate interdependencies between reasoning steps, consistently allocates the same return to any equivalent trajectories, thereby enhancing both stability and effectiveness.

3.4 Optimization

Our objective is to enhance the structured reasoning capabilities of the policy through RL. To alleviate issues of training instability and sample inefficiency in RL (Zhou et al., 2023; Roit et al., 2023), we employ the proximal policy optimization (PPO) algorithm (Schulman et al., 2017) to train the policy π (parameterized by θ), as follows:

$$\mathcal{L}_\pi = \mathbb{E}_t[\min\Big(\frac{\pi'_\theta(a_t|s_t)}{\pi'_{\theta_{old}}(a_t|s_t)}\hat{A}_t, \text{clip}(\frac{\pi'_\theta(a_t|s_t)}{\pi'_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\Big)\hat{A}_t + \beta\mathcal{E}(\pi'_\theta)], \tag{7}$$

where π' represents the probabilities normalized by Equation 2, θ and θ_{old} are parameters of the new and old policies, ϵ is a hyperparameter defining the clipping range, β is the entropy exploration coefficient, and $\mathcal{E}$ is the entropy bonus, which encourages sufficient exploration:

$$\mathcal{E}(\pi'_\theta) = \mathbb{E}_{a_t \sim \pi_\theta}[-\log \pi'_\theta(a_t|s_t)]. \tag{8}$$

Futhermore, $\hat{A}_t$ is the estimate of the advantage function for state s_t, defined as follows:

$$\hat{A}_t = \hat{G}_t - V(s_t). \tag{9}$$

To accurately evaluate return and guide the policy towards better updates, we train the critic by minimizing the difference between its prediction and the TD-target:

$$\mathcal{L}_V = \mathbb{E}_t\left[(V(s_t) - \hat{G}_t)^2\right]. \tag{10}$$

Supervised Warm-up Incorporating the supervised warm-up strategy before RL offers a relatively stable initial policy, which facilitates faster adaptation to the environment, particularly for complex reasoning tasks (Ramamurthy et al., 2023; Wu et al., 2023). Therefore, we convert the structured reasoning into single-step supervised data to warm up the policy as follows:

$$\mathcal{L}_{\text{warmup}} = -\sum_i \log p(y_i|s_t, y_{<i}). \tag{11}$$

where y is the golden action at s_t.

4 Experiments

4.1 Datasets

Tree-structured reasoning We conduct experiments on EntailmentBank (Dalvi et al., 2021), the first dataset that supports structured explanation with entailment trees. Following (Hong et al., 2023), we also conduct experiments on EntailmentBankQA (Tafjord et al., 2022), whose objective is to reach the answer based on the entailment tree.

Graph-structured reasoning We conduct experiments on the STREET benchmark (Ribeiro et al., 2023) to assess the performance of graph-structured reasoning. Please refer to Appendix B for more details about the dataset statistics.

4.2 Baselines

For EntailmentBank, we compare with single-pass methods, such as EntailmentWriter (Dalvi et al., 2021), and step-by-step methods including METGEN (Hong et al., 2022), IRGR (Neves Ribeiro et al., 2022), RLET (Liu et al., 2022), NLProofs (Yang et al., 2022) and FAME (Hong et al., 2023). For EntailmentBankQA, we compare with Selection-Inference (SI) (Creswell and Shanahan, 2022) and FAME (Hong et al., 2023). For the STREET benchmark, we compare with the method proposed in (Ribeiro et al., 2023). Furthermore, we conduct comparisons with GPT-4 (OpenAI, 2023) equipped with Chain-of-Thought (CoT) (Wei et al., 2022), Tree of Thought (ToT) (Yao et al., 2023a) and ReAct (Yao et al., 2023b).

4.3 Implementation Details

For a fair comparison[2], the policy is built with a T5-large model (Raffel et al., 2020), while the critic is the encoder of T5-large combined with a MLP ($\tanh$ as the activation function). For a supervised warm-up, we set a learning rate of 1e-5, a batch size of 16, and train the model for 20 epochs. For RL training, we set learning rate 2e-6 for both policy and critic, discounter factor γ as 0.95, batch size as 3, buffer size as 12, buffer training epochs N_K as 2, ϵ as 0.2, and β as 1e-4. More implementation details can be found in Appendix C.

4.4 Evaluation Metrics

For EntailmentBank, we evaluate T_{pred} with the following dimensions: Leaves, Steps, Intermediates, and Overall AllCorrect. For STREET benchmark, we evaluate the reasoning graphs with two dimensions: Answer Accuracy and Reasoning Graph Accuracy. Note that Overall AllCorrect and Reasoning Graph Accuracy are extremely **strict** metrics, where any deviations will result in a score of 0. More metrics details can be found in Appendix D.

5 Result Analysis

5.1 Structured Reasoning

EntailmentBank As shown in Table 2, our SEER outperforms all baseline methods on the most strict metric, "Overall AllCorrect", across all three tasks. Specifically, our method achieves an absolute improvement of 1.7%/1.4%/1.0% in Task 1/2/3 compared to the strongest baseline. The steps dimension, i.e., premises selection, is the core of EntailmentBank[3], contributing to enhancing the accuracy of both leaves and intermediates dimensions, thereby improving the overall AllCorrect metric. (1) Compared to SOTA supervised methods, such as NLProofs and FAME, our method exhibits significant advantages in the steps dimension. This demonstrates that focusing solely on isolated single-step reasoning through supervised learning may yield suboptimal solutions in intricate structured reasoning tasks, even though employing advanced planning algorithms, such as Monte-Carlo planning in FAME. (2) Compared to the SOTA RL-based method, our method outperforms RLET by 5.8%/9.0%/6.0% in Task 1/2/3. Our method employs a generative model as the policy to circumvent the issue of enumerating actions, facilitating the policy's understanding of structured reasoning tasks (generating potential actions by itself). Moreover, our proposed structure-based return more effectively captures the tree-structured logical dependencies between steps and can assign stable returns for equivalent trajectories, which significantly improves reasoning abilities. Subsequent ablation studies will further demonstrate this. (3) Compared to GPT-4 with CoT, ToT, and ReAct, our method achieves an absolute improvement of 1.9% in Task 3. Although GPT-4 exhibits outstanding reasoning capabilities surpassing many other baselines, its performance relies on a vast number of parameters. Details about the prompts of GPT-4 can be found in Appendix F.

EntailmentBankQA Following Creswell and Shanahan (2022), we introduce the halter module to generate answers based on T_{pred} and substitute hypothesis with question and option during the reasoning process. As illustrated in Table 3, our method surpasses FAME by an absolute margin of 1.2%/7.4% in Task 1/2. While both FAME and SI are supervised methods, FAME significantly outperforms SI by enhancing the model's reasoning and exploration capabilities through Monte-Carlo planning. However, our method enhances the structured reasoning capabilities of the policy rather than focusing solely on single-step reasoning, which can significantly improve the quality of the entailment tree to aid in answering, especially

[2]Previous studies have consistently utilized T5-large as the base model. Despite the existence of more advanced generative models (Du et al., 2022; Touvron et al., 2023), using T5-large enables us to maintain a fair comparison.

[3]A comprehensive error analysis is detailed in Appendix G.

Task	Method	Leaves		Steps		Intermediates		Overall
		F1	AllCorrect	F1	AllCorrect	F1	AllCorrect	AllCorrect
Task1	EntailmentWriter	98.7	84.1	50.0	38.5	67.6	35.9	34.4
	METGEN	**100.0**	**100.0**	57.9	42.1	71.3	39.2	37.0
	IRGR	97.6	89.4	50.2	36.8	62.1	31.8	32.4
	RLET	**100.0**	**100.0**	54.6	40.7	66.9	36.3	34.8
	NLProofS	97.8	90.1	55.6	42.3	**72.4**	40.6	38.9
	SEER (Ours)	**100.0**	**100.0**	**67.6**	**52.6**	70.3	**42.6**	**40.6**
Task2	EntailmentWriter	83.2	35.0	39.5	24.7	62.2	28.2	23.2
	METGEN	83.7	48.6	41.7	30.4	62.7	32.7	28.0
	IRGR	69.9	23.8	30.5	22.3	47.7	26.5	21.8
	RLET	81.0	39.0	38.5	28.4	56.3	28.6	25.7
	NLProofS	**90.3**	**58.8**	47.2	34.4	**70.2**	37.8	33.3
	SEER (Ours)	86.4	53.5	**56.8**	**39.7**	66.3	**38.3**	**34.7**
Task3	EntailmentWriter	35.7	2.9	6.1	2.4	33.4	7.7	2.4
	METGEN	34.8	8.7	9.8	8.6	36.7	20.4	8.6
	IRGR	45.6	11.8	16.1	11.4	38.8	20.9	11.5
	RLET	38.3	9.1	11.5	7.1	34.2	12.1	6.9
	NLProofS	43.2	8.2	11.2	6.9	42.9	17.3	6.9
	FAME	43.4	**13.8**	16.6	12.4	40.6	19.9	11.9
	GPT4-CoT	44.1	12.1	15.4	10.8	43.1	20.6	10.8
	GPT4-ToT	43.3	12.0	15.8	11.0	43.9	20.0	11.0
	GPT4-ReAct	45.8	12.9	14.1	10.5	43.5	**21.5**	10.5
	SEER (Ours)	**47.1**	**13.8**	**17.4**	**12.9**	**45.1**	18.8	**12.9**

Table 2: Experiment results on EntailmentBank. Bold and underlined texts highlight the best method and the runner-up. RLET is based on DeBERTa-large (He et al., 2023), while all other methods are based on T5-large. All baseline results come from published papers. We use the `gpt-4-1106-preview` version for GPT-4.

Method	Task 1	Task 2
SI+Halter	72.4	55.9
SI+Halter+Search	83.2	72.9
FAME	91.5	78.2
SEER (Ours)	**92.7**	**85.6**

Table 3: Experiment results on the EntailmentBankQA. SI is based on Chinchilla-7B (Hoffmann et al., 2022).

Method	SCONE	GSM8K	AQUA-RAT	AR-LSAT
Answer Accuracy				
STREET	69.6	10.4	28.7	28.0
GPT4 †	66.0	**94.0**	**78.0**	32.0
SEER (Ours)	**72.4**	21.4	37.6	**33.5**
Reasoning Graph Accuracy				
STREET	60.0	0.7	0.0	0.0
GPT4 †	32.0	10.0	4.0	2.0
SEER (Ours)	**64.8**	**13.4**	**8.1**	**7.2**

Table 4: Experiment results on STREET benchmark. † indicates we recorded the best results in CoT, ToT, and ReAct for brevity.

in complex reasoning environments.

STREET As shown in Table 4, compared to GPT-4, our method has achieved absolute improvements of 4.8%/3.4%/4.1%/5.2% across various datasets, although the Reasoning Graph Accuracy is a very strict metric (Ribeiro et al., 2023). While GPT-4 excels at answering questions (far surpassing other methods), its parameter is thousands of times greater than other methods. Moreover, during the reasoning process, GPT-4 is prone to hallucinations (Rawte et al., 2023), resulting in poor performance in structured reasoning, particularly evident in the "Reasoning Graph Accuracy" metric. Since SCONE contains sufficient data as well as similar QA and reasoning patterns, we observe that the STREET method would outperform GPT-4 on SCONE. However, by obtaining high-quality reasoning graphs, our method achieves absolute improvements of 2.8%/11.0%/8.9%/5.5% compared to the STREET method, significantly improving answer accuracy and trustworthiness. In reasoning graphs, a state may have multiple parent nodes. Our structure-based return (Equation 6) still precisely describes the cumulative reward for each

Method	eQASC		eOBQA	
	P@1	NDCG	P@1	NDCG
EntailmentWriter	52.48	73.14	69.07	89.05
EntailmentWriter-Iter	52.56	73.28	72.15	90.19
METGEN	55.81	74.19	74.89	90.50
FAME	53.36	79.64	73.09	89.32
GPT-4	54.00	88.82	**85.36**	91.19
SEER (Ours)	**60.33**	**89.76**	77.50	**94.62**

Table 5: Cross-dataset performance on the eQASC and eOBQA.

Method	Leaves	Steps	Intermediates	Overall
SEER (Ours)	**13.8**	**12.9**	**18.8**	**12.9**
w/o redundant	13.2	12.6	18.5	12.3
w/o structure-based return	12.9	11.7	18.5	11.1
w/o RL	10.2	9.4	17.1	9.1

Table 6: Ablation study of each component.

state, thereby facilitating reasoning performance in graph-structured reasoning.

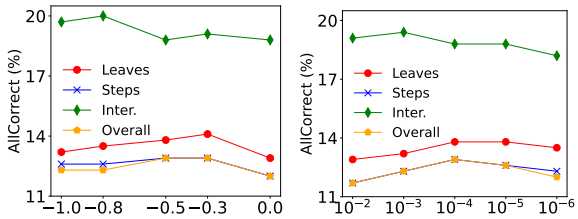
5.2 Cross-dataset Performance

To evaluate the generalization performance, we conduct cross-dataset experiments on eQASC and eOBQA[4] (Jhamtani and Clark, 2020). We apply the policy of Task 2 for selection without training on eQASC or eOBQA. As illustrated in Table 5, our method exhibits significant superiority in cross-dataset generalization. Compared to supervised methods, our SEER, following the inherent structural nature of entailment trees, can better capture the logical dependencies between reasoning steps, which can effectively promote the generalization ability of the policy. The experimental results further validate the effectiveness of our method.

5.3 Ablation Studies

To evaluate the contribution of each component, we conduct extensive ablation studies. As shown in Table 6, we investigate three different variations of SEER in Task 3 of EntailmentBank: (1) **w/o redundant** neglects redundant steps by assigning a reward of -1. (2) **w/o structure-based return** removes the structure-based return and calculates it using the chained sum of rewards (Equation 5). (3) **w/o RL** removes the RL phase, relying solely on supervised warm-up. We discover that overlooking redundant steps may potentially inhibit the exploration of policy, leading to a performance decline. In addition, the results shown in Table 6 also

[4]More details about the setting of eQASC and eOBQA can be found in Appendix B.



(a) $r_{\text{redundant}}$ (b) β

Figure 3: Parameter sensitivity analysis.

demonstrate that removing the structure-based return severely affects the performance. It not only adequately addresses the equivalent trajectory problems, but also elegantly captures the logical relationships inherent in entailment trees, which is crucial for structured reasoning. Furthermore, it can be seen that removing the RL phase reduces performance by 3.8% of Overall Allcorrect, which is a significant impact for this strict metric. This indicates that relying solely on supervised learning may overlook the logical relationships in structured reasoning, thereby falling into suboptimal solutions.

5.4 Parameter Sensitivity Analysis

As illustrated in Figure 3, we further investigate the impact of $r_{\text{redundant}}$ and β on the performance in Task 3. We observe that compared to treating redundant and erroneous steps equally ($r_{\text{redundant}} = -1$), not penalizing ($r_{\text{redundant}} = 0$) may have more detrimental effects, which allows for unrestricted exploration. Moreover, a suitable β (the coefficient of entropy bonus) is crucial for performance enhancement, as it encourages the policy to break away from the "stereotypes" of supervised warm-up.

6 Conclusions

We propose SEER, a novel approach that facilitates structured reasoning and explanation via RL. To our knowledge, SEER is the first general framework capable of enhancing chained, tree-based, and graph-based structured reasoning. Our structure-based return precisely delineates the hierarchical and branching structure inherent in structured reasoning, effectively facilitating reasoning ability. Furthermore, SEER employs a generative model to represent the policy and refines the reward function, ingeniously circumventing the limitations of existing works. Comprehensive experimental results demonstrate that SEER significantly outperforms state-of-the-art methods and exhibits outstanding cross-dataset generalization performance.

Limitations

Although our method has achieved excellent performance in structured reasoning and explanation, there remains one issue that deserves further exploration for future work: how to perform structured reasoning in the context of multimodal data. This includes combining content from images, tables, or audio data, a form of multimodal structured reasoning that is increasingly prevalent and demanding in real-world scenarios. In future work, we plan to extend our SEER to accommodate multimodal scenarios.

Ethics Statement

This work focuses primarily on structured reasoning and explanation problems, and its contributions are entirely methodological. Therefore, this work does not have direct negative social impacts. For the experiments, we have open-sourced the code and utilized openly available datasets commonly used in previous research, without any sensitive information to our knowledge. The authors of this work adhere to the ACL ethical guidelines, and the application of this work does not present any apparent issues that may lead to ethical risks.

Acknowledgements

This work is supported by the National Key R&D Program of China (NO.2022ZD0160102). Chao Yang is supported by the Shanghai Post-doctoral Excellent Program (Grant No. 2022234).

References

Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. 2023. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In *ACL*.

Richard Bellman. 1957. A markovian decision process. *Journal of mathematics and mechanics*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Guoxin Chen, Yiming Qian, Bowen Wang, and Liangzhi Li. 2023. MPrompt: Exploring multi-level prompt tuning for machine reading comprehension. In *Findings of EMNLP*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*.

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

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *CoRR*.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *EMNLP*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *ACL*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *ACL*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. In *ICLR*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*.

Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. 2022. METGEN: A module-based entailment tree generation framework for answer explanation. In *Findings of NAACL*.

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with Monte-Carlo planning. In *ACL*.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *EMNLP*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. *AAAI*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *ECML*. Springer.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2021. Qed: A framework and dataset for explanations in question answering. *Transactions of the Association for computational Linguistics*.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *NeurIPS*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2022. RLET: A reinforcement learning based approach for explainable QA with entailment trees. In *EMNLP*.

Yajiao Liu, Xin Jiang, Yichun Yin, Yasheng Wang, Fei Mi, Qun Liu, Xiang Wan, and Benyou Wang. 2023. One cannot stand for everyone! leveraging multiple user simulators to train task-oriented dialogue systems. In *ACL*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *ACL*.

John McCarthy. 1959. Programs with common sense.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchi Chen, Peng Xu, Zhiheng Huang, Andrew Arnold, and Dan Roth. 2022. Entailment tree explanations via iterative retrieval-generation reasoner. In *Findings of NAACL*.

OpenAI. 2023. GPT-4 technical report.

Gabriel Poesia, WenXin Dong, and Noah Goodman. 2021. Contrastive reinforcement learning of symbolic reasoning domains. In *NeurIPS*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *ACL*.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *ICLR*.

Vipula Rawte, Amit P. Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *CoRR*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henghui Zhu, Rui Dong, Deguang Kong, Juliette Burger, Anjelica Ramos, Zhiheng Huang, William Yang Wang, George Karypis, Bing Xiang, and Dan Roth. 2023. STREET: A multi-task structured reasoning and explanation benchmark. In *ICLR*.

Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Leonard Hussenot, Orgad Keller, Nikola Momchev, Sabela Ramos Garea, Piotr Stanczyk, Nino Vieillard, Olivier Bachem, Gal Elidan, Avinatan Hassidim, Olivier Pietquin, and Idan Szpektor. 2023. Factually consistent summarization via reinforcement learning with textual entailment feedback. In *ACL*.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *EMNLP*.

Soumya Sanyal, Harman Singh, and Xiang Ren. 2022. FaiRR: Faithful and robust deductive reasoning over natural language. In *ACL*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *ACL*.

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of ACL-IJCNLP*.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *EMNLP*.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2020. A survey on explainability in machine reading comprehension. *CoRR*.

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

Marco Valentino, Mokanarangan Thayaparan, and André Freitas. 2021. Unification-based reconstruction of multi-hop explanations for science questions. In *EACL*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *NeurIPS*.

Anne Wu, Kiante Brantley, Noriyuki Kojima, and Yoav Artzi. 2023. lilGym: Natural language visual reasoning with reinforcement learning. In *ACL*.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *LREC*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2023. Are large language models really good logical reasoners? a comprehensive evaluation from deductive, inductive and abductive views. *arXiv preprint arXiv:2306.09841*.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *EMNLP*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *ICLR*.

Fei Yu, Hongbo Zhang, and Benyou Wang. 2023. Nature language reasoning, a survey. *arXiv preprint arXiv:2303.14725*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2021. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*.

Jinfeng Zhou, Zhuang Chen, Bo Wang, and Minlie Huang. 2023. Facilitating multi-turn emotional support conversation with positive emotion elicitation: A reinforcement learning approach. In *ACL*.

A Algorithm Details

Algorithm 1: The training process of SEER

Input: Structured reasoning dataset $\mathcal{D}$; Training epochs N_{warmup}, N and N_K; batch size b_{warmup} and b_{mini};
Output: The optimal parameter of policy

```
/* (1) Supervised Warm-up phase */
```
1 Initialise policy parameters π_θ
2 Convert $\mathcal{D}$ into single-step data $\mathcal{D}_{step}$
3 **for** *epoch* $= 1$ *to* N_{warmup} **do**
4 **for** $i = 1$ *to* $|\mathcal{D}_{step}|/b_{warmup}$ **do**
5 sample minibatch from $\mathcal{D}_{\text{step}}$
6 update parameters π_θ by Eq. 11

```
/* (2) RL phase */
```
7 Initialize the critic parameters V
8 **for** *epoch* $= 1$ *to* N **do**
9 Initialise training buffer $\mathcal{B} \leftarrow \varnothing$
  `// Filling the replay buffer`
10 **while** $\mathcal{B}$ *not full* **do**
11 sample $\{h, X, T_{\text{gold}}\}$ from $\mathcal{D}$
12 collect trajectory τ via π_θ
13 assign a reward for each step in τ
14 fill buffer $\mathcal{B}$ with $\{s_t, a_t, r_t\}$ from τ
  `// Performing k-epoch updates per buffer`
15 **for** $epoch_k = 1$ *to* N_K **do**
16 sample $\{(s_t, a_t, r_t)\}_{b_{\text{mini}}}$ from $\mathcal{B}$
17 compute $\mathcal{E}(\pi'_\theta)$ and $\hat{A}_t$ by Eqs. 8, 9
18 update policy π_θ by Eq. 7
19 update critic V by Eq. 10

Algorithm 1 describes the overall training process of our proposed SEER in detail, which primarily consists of two phases: supervised warm-up and reinforcement learning (RL). In the supervised warm-up phase, the structured reasoning is first decomposed into single-step reasoning data (Line 2). Then, we employs supervised learning to guide the policy π_θ to quickly adapt to the complex reasoning environments (Lines 3-6). This is particularly beneficial when the number of parameters in the policy is relatively small (Akyurek et al., 2023; Liu et al., 2023). In the RL phase, we initially populate the replay buffer $\mathcal{B}$ through the policy π_θ (Lines 9-13). Then, we update the parameters of the policy and the critic using the buffer data. To improve sample efficiency, N_K updates are performed for each replay buffer (Lines 14-18).

For the inference process, we only need to use the policy (without the critic) for structured reasoning. Specifically, as illustrated in the trajectory rollout of Figure 2, we update the state by the policy and the entailment module. Then, we end the reasoning process until the stopping criteria are satisfied. Finally, we backtrack to construct the entire structured explanation, taking the last intermediate conclusion as the hypothesis for entailment tree (or the answer for the STREET benchmark).

B Datasets Details

Datasets of Structured Reasoning Table 7 describes the statistics of datasets in detail. In the answer types, "K-Way MC" stands for multiple choice answer with K options.

EntailmentBank (Dalvi et al., 2021) comprises 1,840 expert-annotated entailment trees with an average of 7.6 nodes spanning across 3.2 entailment steps. The facts are derived from the WorldTree V2 corpus (Xie et al., 2020). Based on different facts X, there are three progressively more challenging tasks: **Task1 (no-distractor)**, **Task2 (distractor)** and **Task3 (full-corpus)**. For GPT-4, we employ all the data in Task 3 from EntailmentBank to evaluate its performance. EntailmentBank was originally designed for post-hoc tree reconstruction tasks instead of QA, Tafjord et al. (2022) converted it into EntailmentBankQA where the task is to choose the correct answer given multiple choice options rather than deriving hypothesis h.

To construct the STREET benchmark, Ribeiro et al. (2023) standardized many QA datasets, such as ARC (Clark et al., 2018), SCONE (Long et al., 2016), GSM8K (Cobbe et al., 2021), AQUA-RAT (Ling et al., 2017) and AR-LSAT (Zhong et al., 2021), in the graph-structured explanation format, where the tasks are converted into answering the question based on the predicted reasoning graphs. Please note that ARC in STREET is congruent with Task 1 of EntailmentBankQA (Ribeiro et al., 2023), hence, we do not repeat the experiment for this task in Table 4. Due to the high cost of GPT-4, we randomly sample 50 instances from each dataset in the STREET benchmark to evaluate GPT-4's performance.

Datasets of Cross-dataset Experiments To evaluate the generalization performance of our method, following Hong et al. (2022), we conduct cross-dataset experiments on eQASC and eOBQA (Jhamtani and Clark, 2020), which collect *one-step* entail-

Task Name	Task Domain	# Questions	# Reasoning Steps	Reasoning Type	Answer Type
EntailmentBank	Science	1,840	5,881	Tree-structured	/
EntailmentBankQA (ARC)	Science	1,840	5,881	Tree-structured	4-Way MC
SCONE	Processes	14,574	130,482	Graph-structured	State Pred.
GSM8K	Math	1,030	4,666	Graph-structured	Number
AQUA-RAT	Math	1,152	7,179	Graph-structured	5-Way MC
AR-LSAT	Logic	500	2,885	Graph-structured	5-Way MC

Table 7: Datasets Statistics of Structured Reasoning.

ment trees for questions from QASC (Khot et al., 2020) and OpenBookQA (Mihaylov et al., 2018), respectively. The goal of this task is to select valid one-step trees from the candidate set and evaluate the results with P@1 and NDCG metrics (Hong et al., 2022). Questions with no valid tree are filtered. The candidate sets for eQASC and eOBQA are composed of 10 and 3 sentences respectively.

C Implementation Details

C.1 Stopping criteria

For a fair comparison, we use the T5-large model to represent the policy. However, we observe that T5-large tends to perform "Reason" actions more frequently, which is caused by the smaller number of model parameters and the issue of having only a few "End" instances. Moreover, unlike GPT-4, T5-large is less able to recognize when a hypothesis has been inferred and when to stop. Therefore, we attach two extra stopping criteria in addition to the "End" action: (1) The semantic similarity between the intermediate conclusion and the hypothesis exceeds a predefined threshold, i.e., BLEURT(i_*, h) > 1. (2) Exceeding the maximum number of reasoning steps (set to 20 in this paper).

C.2 Alignment algorithm

Following (Dalvi et al., 2021), we evaluate the intermediate steps based on Jaccard similarity. Specifically, the intermediate nodes i_* in T_{pred} are aligned with the intermediate nodes in T_{gold} that have the maximum Jaccard similarity. If the Jaccard similarity between the intermediate node in T_{pred} and all intermediate nodes in T_{gold} is zero, it is aligned with "NULL". Note that only the intermediate node that is perfectly matched with a node in T_{gold}, i.e., the Jaccard similarity is 1, is considered as a correct step. Figure 5 provides a detailed illustration of this process. The alignment process is similar in the reasoning graphs (Ribeiro et al., 2023).

C.3 Retriever for Task 3

In Task 3 of EntailmentBank, first, it is necessary to retrieve relevant sentences from the corpus (Dalvi et al., 2021). The research focus of this paper is to enhance the structured reasoning ability of the policy. Therefore, we directly adopt the retriever and its associated parameters proposed in previous work (Hong et al., 2023), which is based on the all-mpnet-base-v2 model (Reimers and Gurevych, 2019). For a fair comparison, we retrieve the top 25 most relevant sentences as X for Task 3.

C.4 Entailment Module

The entailment module is also based on the T5-large model, taking premises as input and generating intermediate conclusions. Our primary focus is to enhance the structured reasoning ability of the policy through RL, therefore, we directly employ the entailment module that has already been trained in previous work (Hong et al., 2023), which also aids in a fair comparison.

C.5 Halter Module

In EntailmentBankQA, we employ the Halter module (Creswell and Shanahan, 2022) to answer questions based on the predicted entailment trees. In this paper, the Halter module is built upon the T5-large model. The module is trained with a learning rate of 1e-5 and a batch size of 16.

C.6 Entailment Tree Construction

To evaluate the correctness of each reasoning step, we have to reconstruct the trajectory into an entailment tree T_{pred} and compare it with T_{gold}. Figure 5 illustrates this reconstruction process. We consider the last intermediate conclusion as the hypothesis and then construct the predicted entailment tree based on the reasoning relationship of each step. The reconstruction process is similar in the reasoning graphs (Ribeiro et al., 2023).

C.7 Running time

In our experimental setting, the average training time per entailment tree in SEER is 6.98 seconds, and the average inference time per entailment tree in SEER is 3.91 seconds. As reported in their papers, the inference time per entailment tree in RLET (Liu et al., 2022) and FAME (Hong et al., 2023) are 9.34 seconds and 30.77 seconds, respectively. FAME leverages Monte-Carlo planning, necessitating the exploration of numerous nodes to enhance the reasoning capability of the policy, thus requiring considerable computational time. Our proposed SEER significantly surpasses FAME in terms of both efficiency and effectiveness.

C.8 Experiment Environments

All experiments were conducted on Ubuntu 22.04 equipped with NVIDIA A100 GPUs. Our code mainly depends on python 3.10[5] and PyTorch 2.0.1[6]. The pre-trained language models are derived from `HuggingFace Transformers`[7].

C.9 Details of Reasoning Graphs

For the reasoning graphs in the STREET Benchmark, the implementation details are slightly different from the entailment trees. In the reasoning graphs, reasoning steps may possess multiple parent nodes, and a fact (x_*) or intermediate conclusion (i_*) may be utilized multiple times (Ribeiro et al., 2023). Therefore, in the reasoning graph, we refrain from incorporating previously used premises into U_t, instead continually expanding the candidate sentence set C_t through newly derived intermediate conclusions. In other words, the state in the reasoning graphs is updated according to the following rules: $P_{t+1} = P_t \cup \{\text{<premises>} \rightarrow i_t\}$, $C_{t+1} = \{X \cup I_{t+1}\}$, and $I_{t+1} = I_t \cup \{i_t\}$.

C.10 Other Implementation Details

For GPT-4, we set the temperature to 0.7. For Tree of Thought, we set $b = 5$ candidates at each step, and then vote to select the optimal action. Details regarding the prompts of CoT, ToT, and ReAct can be found in Appendix F. For all baselines, we obtain the optimal results based on experimental results or hyperparameter settings derived from the original papers. For our method, we initialize the critic with the encoder of the warm-up policy to expedite the convergence of the critic and facilitate policy updates. The hidden layer dimension of the MLP in the critic is set to 512.

[5] `https://www.python.org/`
[6] `https://pytorch.org/`
[7] `https://huggingface.co/`

D Metrics Details

For EntailmentBank, we follow (Dalvi et al., 2021) and evaluate the entailment tree T_{pred} using three dimensions:

• **Leaves**: To evaluate the leaf nodes of T_{pred}, we compute F1 by comparing X_{pred} with X_{gold}.

• **Steps**: To evaluate the structural correctness of each step, we compare all steps between T_{pred} and T_{gold} and then compute F1. A predicted step is considered structurally correct if its premises (e.g., x_*, i_*) exactly match the gold premises.

• **Intermediates**: To evaluate the intermediate conclusions, we compare the aligned intermediate conclusions and then compute F1. A predicted intermediate conclusion is deemed correct if the BLEURT score (Sellam et al., 2020) exceeds 0.28.

For each dimension, the AllCorrect score is 1 if F1 is 1, otherwise 0. Given the above scores, we employ the **Overall AllCorrect** metric to comprehensively evaluate T_{pred}, which takes a value of 1 if and only if all leaves, steps, and intermediates are correct. Note that this is an extremely **strict** metric, where any deviations in T_{pred} will result in a score of 0.

For the STREET benchmark, we follow (Ribeiro et al., 2023) and adopt two metrics, namely, the answer to the question and the quality of the reasoning graphs, to evaluate different methods.

• **Answer Accuracy:** This metric measures the ability to correctly answer questions. The answer accuracy serves as an upper bound for other metrics, as any reasoning graph generated with incorrect answers is also labeled as incorrect.

• **Reasoning Graph Accuracy:** This metric compares the predicted reasoning graph and the golden reasoning graph from the aspects of the graph structure and the content of intermediate conclusions. Please note that this is a stringent metric, with minor deviations from the golden reasoning graph resulting in the prediction being incorrect.

E Illustrations and Case Study of SEER

Given a hypothesis h and initial facts X, we first obtain the trajectory through the reasoning process, as shown in Figure 4. The state update follows the Markov decision process (Bellman, 1957), meaning the current state only depends on the previous

state. Figure 4 is an erroneous reasoning example to better illustrate the following steps. Then, we convert the trajectory τ into an entailment tree T_{pred} and align it with T_{gold} to assign reward for each intermediate conclusion (as presented in Figure 5). Furthermore, Figure 6 elucidates the issue of equivalent trajectories, and previous work can not accurately describe the logical relationship between different states in entailment trees.

F Prompts for GPT-4

Figures 7 and 8 show the Chain-of-Thought (CoT) (Wei et al., 2022) and ReAct (Yao et al., 2023b) prompts for GPT-4, and figures 9 and 10 show the prompts of thought generator and state evaluator in Tree of Thought (ToT) (Yao et al., 2023a), respectively. We provide a detailed introduction to the task definition and guide the model to respond in the required format. We randomly selected three examples for in-context learning. For a fair comparison, we use the same examples for CoT and ReAct, attributing similar thoughts to them. ReAct divides the dialogue into two rounds, "Thought" and "Action", to query GPT-4. For ToT, following (Yao et al., 2023a), we generate candidate actions using a thought generator and subsequently select and execute the optimal action through a state evaluator. Due to its exceptional reasoning capabilities and self-evaluation strategy, ToT achieves superior results compared to CoT and ReAct, as shown in Table 2. However, ToT requires higher costs in comparison to CoT and ReAct.

G Error Analysis

We conduct a comprehensive error analysis on Task2 and Task3 of EntailmentBank.

G.1 Error Analysis of Task2

We randomly sample 50 entailment trees where SEER made incorrect reasoning. We find the following four types of errors.

(1) Reasoning Step Error (62%). This is the main source of errors and predominantly depends on whether the policy can select the correct premise. We observe that a small portion of the errors (accounting for 12.9% of this error type) use all the gold leaves, but have errors in the combination order. Compared to other reasoning step errors, the policy identified the correct premise. For example, the gold steps are "x_{24} & $x_5 \rightarrow i_1$; i_1 & $x_{23} \rightarrow h$" and the error steps predicted by SEER are "x_{23} & $x_5 \rightarrow i_1$; i_1 & $x_{24} \rightarrow i_2$".

(2) Early Termination Error (18%). We observe that the reasoning process may terminate prematurely and the existing entailment steps are all correct. On one hand, T5-large outputs "End" prematurely, unlike GPT-4 which can accurately judge when to stop. On the other hand, the entailment module might erroneously infer a hypothesis, leading to premature termination.

(3) Intermediate Conclusion Error (10%). This error type is different from the above error (where the entailment module prematurely infers a hypothesis). Intermediate conclusion error denotes errors triggered by incorrect entailment in the intermediate conclusions, despite having correct leaves and steps. For a fair comparison, we used the entailment module that has already been trained in previous work (Hong et al., 2023). It is noted that the reasoning part, which is the focus of our paper, is completely correct in this type of error, and this type of error can be mitigated by training a better entailment module.

(4) Imperfect Evaluation (10%). We discover that some trees deemed as invalid are valid in fact, indicating that current automated metrics underestimate the validity of the trees. The most common reason is that there are multiple valid ways to construct an entailment tree. For example, consider the structure of a gold tree: "x_1 & x_2 & $x_3 \rightarrow h$" may be predicted as: "x_1 & $x_2 \rightarrow i_1$; i_1 & $x_3 \rightarrow i_2$".

G.2 Error Analysis of Task3

Task 3 requires retrieving an initial set of facts X from the corpus. Therefore, in addition to the errors in Task 2 described above, we found that Task 3 has its own unique set of errors.

(1) Missing Gold Leaves Error (58%). Missing gold leaves error refers to the case where the gold leaves are not included in the facts X retrieved from the corpus. This case will inevitably lead to an error in the predicted entailment tree, regardless of how powerful the policy is. The bottleneck of this error lies in the retrieval model. For a fair comparison, we directly use the retrieval model provided in previous work (Hong et al., 2023).

(2) Reasoning Errors (42%). The four error types described in G.1 account for 42% in Task3.

We also discovered that the reasoning graph contains similar error types as found in entailment trees, as they both belong to structured reasoning.

Question: Melinda learned that days in some seasons have more daylight hours than in other seasons. Which season receives the most hours of sunlight in the Northern Hemisphere?

Answer: summer

Hypothesis h: northern hemisphere will have the most sunlight in summer

s_1: h; $P_1 = \emptyset$; $C_1 = \{$

x_1: when a hemisphere is tilted towards the sun, that hemisphere receives more direct sunlight
x_2: if a place is in summer, then it will have the most sunlight
x_3: summer has the most sunlight
x_4: the northern hemisphere is a kind of hemisphere of earth
x_5: the equator receives the most amount of direct sunlight throughout the year
x_6: the amount of daylight is greatest in the summer
x_7: when a hemisphere is tilted away from the sun, that hemisphere receives less direct sunlight
x_8: the amount of daylight is greatest on the summer solstice
x_9: the northern hemisphere is tilted toward the sun
x_{10}: june is during the summer in the northern hemisphere
x_{11}: the poles receives the least amount of direct sunlight throughout the year
x_{12}: a hemisphere is a part of earth
x_{13}: summer is when a hemisphere is tilted towards the sun
x_{14}: the western hemisphere is a kind of hemisphere of earth
x_{15}: united states is located in the northern hemisphere
x_{16}: north america is located in the northern hemisphere
x_{17}: the north pole is located in the northern hemisphere
x_{18}: sint eustatius (netherlands) is located in the northern hemisphere
x_{19}: china is located in the northern hemisphere
x_{20}: winter has the least sunlight
x_{21}: the amount of daylight is least on the winter solstice
x_{22}: the amount of daylight is least in the winter
x_{23}: netherlands is located in the northern hemisphere
x_{24}: the southern hemisphere is a kind of hemisphere of earth
x_{25}: a hemisphere of earth is a kind of place$\}$

a_1: Reason: x_4 & x_{25}
Entailment Module: x_4 & $x_{25} \rightarrow i_1$: the northern hemisphere is a kind of place.

s_2: h; $P_2 = \{x_4 \,\&\, x_{25} \rightarrow i_1\}$; $C_2 = \{$

i_1: the northern hemisphere is a kind of place.
...
x_{10}: june is during the summer in the northern hemisphere
x_{13}: summer is when a hemisphere is tilted towards the sun
...
x_{24}: the southern hemisphere is a kind of hemisphere of earth$\}$

a_2: Reason: x_{10} & x_{13}
Entailment Module: x_{10} & $x_{13} \rightarrow i_2$: june is when the northern hemisphere tilts toward the sun.

... s_3 ... a_3 ...

s_4: h; $P_4 = \{x_4 \,\&\, x_{25} \rightarrow i_1;\ x_{10} \,\&\, x_{13} \rightarrow i_2; x_3 \,\&\, x_{17} \rightarrow i_3\}$; $C_4 = \{$

i_1: the northern hemisphere is a kind of place.
i_2: june is when the northern hemisphere tilts toward the sun.
i_3: the north pole experiences the most sunlight during summer.
...
x_{11}: the poles receives the least amount of direct sunlight throughout the year
x_{12}: a hemisphere is a part of earth
x_{14}: the western hemisphere is a kind of hemisphere of earth
x_{15}: united states is located in the northern hemisphere
...
x_{24}: the southern hemisphere is a kind of hemisphere of earth$\}$

a_4: Reason: i_1 & i_2
Entailment Module: i_1 & $i_2 \rightarrow i_4$: the northern hemisphere tilts toward the sun in june.

s_5: h; $P_5 = \{x_4 \,\&\, x_{25} \rightarrow i_1;\ x_{10} \,\&\, x_{13} \rightarrow i_2; x_3 \,\&\, x_{17} \rightarrow i_3;\ i_1 \,\&\, i_2 \rightarrow i_4\}$; $C_5 = \{$

i_3: the north pole experiences the most sunlight during summer.
i_4: the northern hemisphere tilts toward the sun in june.
...
x_{11}: the poles receives the least amount of direct sunlight throughout the year
x_{12}: a hemisphere is a part of earth
...
x_{24}: the southern hemisphere is a kind of hemisphere of earth$\}$

a_5: End
Output: trajectory $s_1(i_1) \rightarrow s_2(i_2) \rightarrow s_3(i_3) \rightarrow s_4(i_4)$

Figure 4: An illustration of the reasoning process of SEER. Note that a_1 is a correct step, a_2 and a_4 are erroneous steps, and a_3 is a redundant step. We start from the initial state s_1 where existing entailment steps $P_1 = \varnothing$ and candidate sentences $C_1 = X$. In each step, we sample an action and update the state until the reasoning is done. For the "Reason" action, we sent the premises to the entailment module. The new conclusion is added to the C, the premises is removed from C and the entailment step is added to the P. For the "End" action, we end the reasoning process and output the trajectory.

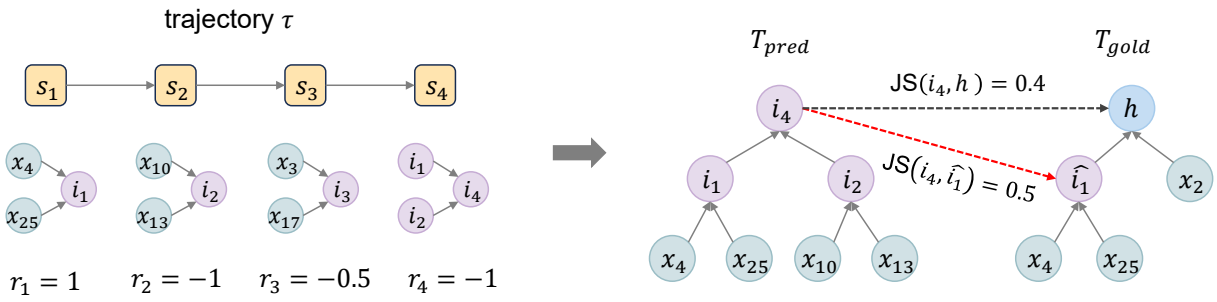


Figure 5: An illustration of the reward and alignment process of SEER. Each reasoning step is a subtree (similarly, each reasoning step is a subgraph in the reasoning graph (Ribeiro et al., 2023)). (1) We construct T_{pred} using the last intermediate conclusion (i_4 in this example) as the hypothesis. (2) We calculate the Jaccard similarity between the intermediate node (i_*) in T_{pred} and each golden intermediate node in T_{gold} ($\hat{i}_1$ and h in this example), and align with the maximum Jaccard similarity. In this example, i_1 is aligned with $\hat{i}_1$ due to $\mathrm{JS}(i_1, \hat{i}_1) = 1$. i_2 is aligned with "NULL". i_4 is aligned with $\hat{i}_1$ due to $\mathrm{JS}(i_4, \hat{i}_1) = 0.5$ and $\mathrm{JS}(i_4, h) = 0.4$. (3) We assign rewards based on the alignment results. Note that i_3 (s_3) is a redundant step. $r_1 = 1$, $r_2 = -1$, $r_3 = -0.5$, and $r_4 = -1$. The reward for each state originates from the tree structure rather than the chained trajectory. Therefore, the return of each state should also follow the tree structure (or graph structure in reasoning graphs) rather than the chained trajectory.

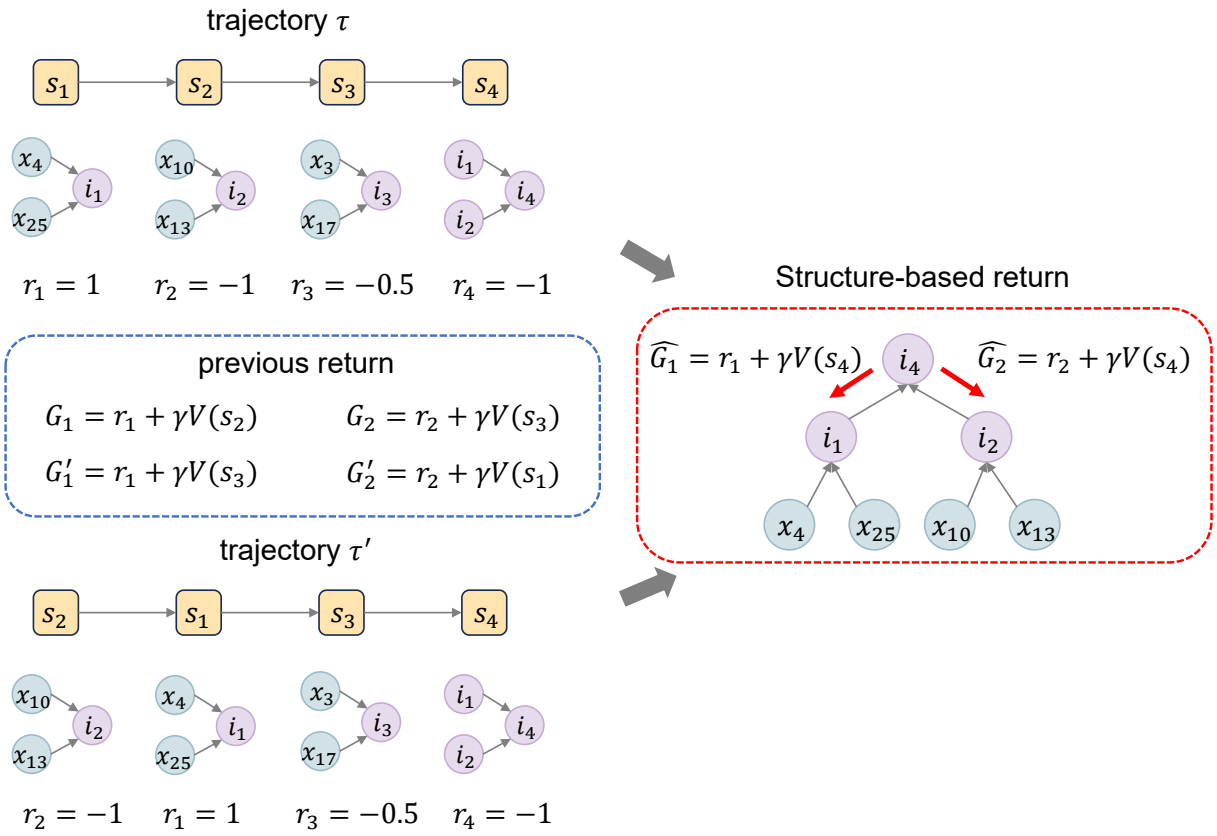


Figure 6: An illustration of the equivalent trajectory and the definition of return. As the reasoning steps of s_1, s_2, and s_3 are mutually independent, the execution order among these steps can be arbitrary. Thus, τ and τ' are equivalent trajectories because they can be converted to the same entailment tree (Dalvi et al., 2021). As shown in blue box, previous work (Liu et al., 2022) defines the return (a.k.a cumulative reward) in a chained trajectory and would assign different returns to s_1 and s_2 in these equivalent trajectories. In contrast, as shown in red box, our structure-based return is defined based on the tree or graph structure inherent in structured reasoning, which is the same source of rewards. Our structure-based return will consistently allocate stable returns to equivalent trajectories, thereby promoting training stability and convergence. Furthermore, maintaining consistency between the sources of rewards and returns can significantly enhance the effectiveness of the policy.

As a quality assurance reasoning system, you first analyze how to perform the reasoning and then immediately give the results of the reasoning. I will give you a hypothesis and a context. I want you to show a reasoning step that goes from some or all of the sentences in the context to the hypothesis. The inference step uses two sentences in the context as premises and obtains a new conclusion. The conclusion should be a valid entailment of the premises. You don't have to choose all the sentences in the context, just the ones you think are sufficient and necessary as premises for your reasoning.

Here are some examples:
Example 1:
Hypothesis: a line graph an be used to show the data of the growth of the vine over a period of time
Context:
x1: length is a measure of distance from one end of an object to the other end of that object
x2: 1 month is equal to 28-31 days
…
x25: a student wants to record the data of the growth of a vine over a period of a day

Your response should be in the following format.
From sentences x23 and x25, we can infer that the student wants to record the data of the growth of a vine over a period of time (labeled as i1). Then, we can conbine this intermediate conclusion (i1) with x5 to derive the hypothesis, due to the x5 describe the line graph is used for showing change.
Proof: x23 & x25 -> i1: the student wants to record the data of the growth of a vine over a period of time; i1 & x5 -> hypothesis;

Example 2:
Hypothesis: the star cluster that captured by the space telescope is a galaxy
Context:
x1: if something is a part of something then that something can be found in that something
x2: distant means great in distance
…
x25: the properties of something can be used to identify / used to describe that something

Your response should be in the following format.
From sentences x8 and x16, we can infer that the star cluster captured by the space telescope, which contains billions of stars, is likely to be a galaxy (referred to as i1). The final hypothesis can be directly derived from this intermediate conclusion i1.
Proof: x16 & x8 -> hypothesis;

Example 3:
Hypothesis:
the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year
Context:
x1: leo is a kind of constellation
x2: to be found in means to be contained in
…
x25: the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year

Your response should be in the following format.
From sentences x1 and x17, we know that Leo is a type of constellation and a constellation contains stars. This leads us to the intermediary conclusion (i1) that Leo contains stars. Combining this intermediate conclusion i1 with x25, which states that the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year, we can infer the hypothesis that the earth revolving around the sun causes Leo to appear in different areas in the sky at different times of year.
Proof: x1 & x17 -> i1: leo is a constellation containing stars; i1 & x25 -> hypothesis;
(END OF EXAMPLES)

Please pay attention to the output format and take care that the reasoning process is as concise as possible without unnecessary steps.

Now reason about the following:

Figure 7: A Chain-of-Thought prompt for GPT-4.

As a quality assurance reasoning system that solves sequential reasoning tasks through interleaving Thought, Action. I will give you a hypothesis and a context. I want you to show a reasoning step that goes from some or all of the sentences in the context to the hypothesis. The inference step uses two sentences in the context as premises and obtains a new conclusion. The conclusion should be a valid entailment of the premises. You don't have to choose all the sentences in the context, just the ones you think are sufficient and necessary as premises for your reasoning.

Here are some examples:
Example 1:
Hypothesis: a line graph an be used to show the data of the growth of the vine over a period of time
Context:
x1: length is a measure of distance from one end of an object to the other end of that object
x2: 1 month is equal to 28-31 days
x3: days (d) are a metric unit used for measuring time generally used for values between 1 and 365
…
x25: a student wants to record the data of the growth of a vine over a period of a day

Thought: From sentences x23 and x25, we can infer that the student wants to record the data of the growth of a vine over a period of time (labeled as i1). Then, we can conbine this intermediate conclusion (i1) with x5 to derive the hypothesis, due to the x5 describe the line graph is used for showing change.
Action: x23 & x25 -> i1: the student wants to record the data of the growth of a vine over a period of time; i1 & x5 -> hypothesis;

Example 2:
Hypothesis: the star cluster that captured by the space telescope is a galaxy
Context:
x1: if something is a part of something then that something can be found in that something
x2: distant means great in distance
x3: discovering something usually requires seeing that something
…
x25: the properties of something can be used to identify / used to describe that something

Thought: From sentences x8 and x16, we can infer that the star cluster captured by the space telescope, which contains billions of stars, is likely to be a galaxy (referred to as i1). The final hypothesis can be directly derived from this intermediate conclusion i1.
Action: x16 & x8 -> hypothesis;

Example 3:
Hypothesis:
the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year
Context:
x1: leo is a kind of constellation
x2: to be found in means to be contained in
x3: move around means revolve
…
x25: the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year

Thought: From sentences x1 and x17, we know that Leo is a type of constellation and a constellation contains stars. This leads us to the intermediary conclusion (i1) that Leo contains stars. Combining this intermediate conclusion i1 with x25, which states that the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year, we can infer the hypothesis that the earth revolving around the sun causes Leo to appear in different areas in the sky at different times of year.
Action: x1 & x17 -> i1: leo is a constellation containing stars; i1 & x25 -> hypothesis;
(END OF EXAMPLES)

Please pay attention to the output format and take care that the reasoning process is as concise as possible without unnecessary steps.

Now reason about the following:

Figure 8: A ReAct prompt for GPT-4. "Thought" and "Action" query GPT-4 in two rounds.

The Prompt of Thought Generator:

As a quality-assured controller in a reasoning system, your primary task is to provide the most insightful action for each step of reasoning towards the hypothesis. Instead of completing the entire reasoning process at once, we want to break it down into multiple steps.

Your candidate actions are:

1. reason: sentX & intX -> new conclusion
2. END

`reason: sentX & intX -> new conclusion` represents the selection of sentX and intX from the current context to derive a new conclusion. Note that only sentences that exist in context can be used, and sentences in Used Premise cannot be used.

`END` indicates that the current hypothesis has been reasoned out and the reasoning can be terminated. Note that the reasoning should end when the conclusion drawn is similar to the hypothesis.

Please give 5 candidate actions that are most likely to facilitate reasoning based on the context.

Note that we are reasoning step by step and only consider the current step, which is the five possible steps to answer the current optimal answer.

Here are some examples:

Example 1:

Hypothesis: the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year

Used Premises:

sent1: leo is a kind of constellation

sent17: a constellation contains stars

Current Proof:

sent1 & sent17 -> int1: leo is a constellation containing stars

Context:

sent8: a motion is a kind of event / action

…

sent25: the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year

int1: leo is a constellation containing stars

Candidate action:

reason: sent11 & sent20 -> planets in the solar system, like earth, orbit the sun

reason: sent8 & sent10 -> motion is an event where an object moves to a direction

reason: sent14 & int1 -> how Leo appears is determined by how its stars look in the sky

reason: sent25 & int1 -> the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year

END

Example 2:

…

Example 3:

…

(END OF EXAMPLES)

Please pay attention to the output format and take care that the reasoning process is as concise as possible without unnecessary steps.

Now reason about the following:

Hypothesis: {hypothesis}

Used Premise:

{used_premise}

Current Proof:

{current_proof}

Context:

{context}

Candidate action:

Figure 9: A Tree of Thought prompt (**Thought Generator**) for GPT-4.

The Prompt of State Evaluator:

Given an instruction and several choices, decide which choice is most promising. Analyze each choice in detail, then conclude in the last line "The best choice is {{s}}", where s the integer id of the choice.

Be careful to follow the output format.

Here are some examples:

Example 1:

Hypothesis: the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year

Used Premises:

sent1: leo is a kind of constellation

sent17: a constellation contains stars

Current Proof:

sent1 & sent17 -> int1: leo is a constellation containing stars

Context:

sent2: to be found in means to be contained in

…

sent25: the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year

int1: leo is a constellation containing stars

Candidate action:

Choice 1:

reason: sent11 & sent20 -> planets in the solar system, like earth, orbit the sun

Choice 2:

reason: sent8 & sent10 -> motion is an event where an object moves to a direction

Choice 3:

reason: sent14 & int1 -> how Leo appears is determined by how its stars look in the sky

Choice 4:

reason: sent25 & int1 -> the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year

Choice 5:

END

Vote: The best choice is 4

Example 2:

…

Example 3:

…

(END OF EXAMPLES)

Now vote about the following:

Hypothesis: {hypothesis}

Used Premise:

{used_premise}

Current Proof:

{current_proof}

Context:

{context}

Candidate action:

{candidate action}

Vote:

Figure 10: A Tree of Thought prompt (**State Evaluator**) for GPT-4.