Reasoning in Reasoning: A Hierarchical Framework for (Better and Faster) Neural Theorem Proving

Anonymous Authors

Abstract

Reasoning is the central objective for artificial intelligence. Autoregressive language models have shown promise in generating reasoning steps token-by-token for problem solving; however, the performance of complex reasoning tasks such as theorem proving is limited by the combinatorially large search spaces and the need for precise decision-making at each step. We develop Reasoning in Reasoning (RiR), a hierarchical framework for reasoning that combines strategic problem decomposition with goal-driven reasoning step generation. Using neural theorem proving as a representative task, our approach breaks down complex theorem problems into smaller, achievable target goals, giving language models: (i) improved generalizability for reasoning step generation, (ii) a more compact and informative search space for reasoning paths, and (iii) an efficient mechanism for learning to plan. Theoretically, we present the rigorous information-theoretic analysis for generalization and efficiency advantage guarantees. Empirically, we show RiR is significantly faster than classical approaches with better performance: RiR improves the state-of-the-art proving accuracy on LeanDojo Benchmark from 50.16% to 53.73%, while significantly reducing the proving time to approximately 1/3 of the original. We believe our framework sets a new direction for goal-driven and information-directed reasoning for language models.

A very powerful approach is to attempt to eliminate everything from the problem except the essentials; that is, cut it down to size. Very often, if you can solve the simple problem, you can add refinements to the solution of this, until you get back to the solution of the one you started with.

– Claude Shannon

1. Introduction

The main question we aim to address in this work is: what would be an effective learning mechanism for language models to solve complex reasoning problems, such as mathematical theorem proving?

We propose an information-theoretic answer, grounded in the principles of goal-driven tree search and problem decomposition, with provable generalization and efficiency guarantees. Specifically, we consider language model reasoners should be trained in a way that is goal-driven and hierarchical, and supervisory signals should take place on different semantic levels during reasoning.

As a background, recent progress in language models have shown promises in generating intermediate reasoning steps for problem solving [Wei et al., 2022], yet the reasoning performance often deteriorates when facing long trajectories or vast spaces where good solutions are sparse. This challenge is particularly evident in automated theorem proving, a task that has been at the core of artificial intelligence research since the field’s early days [Simon, 1969]. The process of crafting a proof is a quintessential example of reasoning [Wang, 1961]. Just as a learning machine aims to generalize from a limited set of training examples to the broader set of all possible observations, such a reasoning agent must navigate from a given set of known theorems and axioms to the vast space of provable statements. An effective strategy from human mathematicians is the decomposition of problems with a sequence of target goals. This approach provides a more informative direction for subsequent reasoning steps, reducing the effective search space, making the task more tractable and interpretable.

We hereby introduce Reasoning in Reasoning (RiR), a simple and fundamental framework for language models to do complex reasoning. Practically in the context of neural theorem proving1, our framework consists of an offline co-training stage, followed by an online goal-driven hierarchical planning stage.

1Another reason for us to choose formal theorem proving as a representative task is its precise state descriptions in the language space. While our framework is general enough for any complex reasoning tasks, we leave the exploration of them to future study.

1Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anonymous@domain.edu>.

In submission to AI for Math Workshop @ ICML 2024.
In the **offline pretraining stage**, we consider the problem of reasoning step generation. Take formal theorem proving as an example, we assume a population distribution on triplets \( D = \{ (s_i, y^*_i, s^*_i) \}_{i=1}^N \) extracted from expert-written proof datasets, where \( s \) denotes the current goal state, \( y^* \) denotes the expert proofstep (tactic), and \( s^* \) denotes the target (next) goal state reached by \( y^* \). Classical approaches [Li et al., 2024; Yang et al., 2023; Han et al., 2021; Azerbayev et al., 2023] consider only next-step proofstep prediction, which trains a proofstep predictor model \( p_\theta(y^* | s) \). We co-train both a goal predictor model \( p_\theta(s^* | s) \) and a goal-driven tactic predictor model \( p_\theta(a^* | s, s^*) \) as shown in Eq. 3. We then provide the first formal guarantee on the generalizability benefits of this co-training intuition, which is referred to as the Co-Training Advantage Theorem as shown in 1.

In the **online planning stage**, we consider the problem of optimal reasoning trajectory planning. Classical approaches uses single-level tree search to traverse the tactic space [Polu and Sutskever, 2020; Yang and Deng, 2019; Lample et al., 2022]. We consider a hierarchical tree search approach as shown in Algorithm 1, where the high-level is for goal search and the low-level is for goal-driven reasoning step search. We provide the first formal guarantee on the sample efficiency benefits of this hierarchical search strategy, which is referred to as the Hierarchical Planning Advantage Theorem as shown in 2.

The new and unique design of RiR is underpinned by a key insight from information theory: by maximizing the mutual information between the environment observations and the reasoning steps, we can encourage the model to generate target goals that creates informative partitioning of the search space. In the context of theorem proving, this objective serves as a principled guide for both the offline co-training and the online hierarchical search. By decomposing complex problems into informative target goals on the high level, RiR enables more efficient exploration of the search space and improved generalization for reasoning.

Our contributions are:

- **Framework**: We develop Reasoning in Reasoning (RiR), a new and general reasoning framework, that practically implements with goal-driven offline pretraining and hierarchical online planning for neural theorem proving.

- **Theory**: We provide the formal guarantees on the generalization and sample efficiency benefits on RiR with a fresh information-theoretic approach, contributing to the Co-Training Advantage Theorem and the Hierarchical Planning Advantage Theorem, respectively.

- **Experiments**: We show that RiR achieves state-of-the-art performance and efficiency on popular benchmark for neural theorem proving including LeanDojo [Yang et al., 2023] and miniF2F [Zheng et al., 2021].

### 2. Preliminaries: Classical Neural Theorem Proving with Language Models

In this section, we focus on basic concepts and classical methods. We present glossary in Table 1. Our new framework will be from Section 3.
The goal is to learn an optimal policy $\pi$ that maximizes the expected cumulative reward over the trajectory. In this case, the optimal policy $\pi^*$ corresponds to the shortest complete proof sequence. Directly learning this policy online can be challenging due to the large search space.

### Neural theorem proving
A neural network parameterized by $\theta$ can act as a policy network that samples single tactics $y_i$ at step $t$. The objective is to find the optimal reasoning trajectory that leads to a solved proof for each query $q$. Let $S$ be state space and $A$ be the action (i.e., tactic) space. This search problem can be formulated as finding a sequence of tactics $y_1, y_2, \ldots, y_T$ such that:

$$s_0 \xrightarrow{y_1} s_1 \xrightarrow{y_2} s_2 \xrightarrow{y_3} \ldots \xrightarrow{y_T} s_T.$$  

The goal is to learn an optimal policy $\pi^*(y_{t+1} | s_t)$ that maximizes the expected cumulative reward over the trajectory. In this case, the optimal policy $\pi^*$ corresponds to the shortest complete proof sequence. Directly learning this policy online can be challenging due to the large search space and sparse rewards in the formal language space, and the problem of automated theorem proving is often tackled via a two-stage framework: (i) offline learning of a tactic generator from expert demonstrations; and (ii) online proof search via the learned tactic generator to find complete proofs by sequentially applying tactics. The following sections describe how classical works do in these two stages.

#### Setup
We frame formal theorem proving as a Markov Decision Process. Starting with a to-prove statement $q$ whose initial state is $s_0$, we sequentially apply tactics $y_i$ to prove it. Under the hood, each tactic applied will make the current state $s_t$ transit to the next state $s_{t+1}$. In the context of theorem proving, the to-prove statement with the proof steps will be deterministically mapped to a state by the environment operator $E(\cdot)$ such that $s_t = E(q \circ y_{1:t})$. Each state is associated with a scalar reward, $r(s_t)$, provided by the environment.

#### Stage 1: offline learning for proofstep generation
Classical approaches [Han et al., 2021; Welleck et al., 2022; Yang et al., 2023; Azerbayev et al., 2023; Li et al., 2024] fine-tune a language model $p_\theta(y^* | s)$ to sample the next proofstep $y$ conditional on current goal $s$. Assume a distribution on pairs $D^{\text{train}}=\{(s_i, y_i^*)\}_{i=1}^N$ extracted from expert datasets, the classical loss function is:

$$L_c(\theta) = -\frac{1}{N} \sum_{(s, y^*) \in D^{\text{train}}} \log p_\theta(y^* | s). \quad (1)$$

The classical prompt format for training and inference is:

\[
\begin{align*}
> \text{Input:} & \quad \{\text{current goal } s\} \\
> \text{Output:} & \quad \{\text{proofstep } y^*\}
\end{align*}
\]

#### Stage 2: online search for complete proof
Classically, given a statement $q$, a full proof $\bar{y}_{1:T}$ is found by constructing a tree [Yang et al., 2023; Azerbayev et al., 2023; Li et al., 2024] with only low-level tactic search. A common choice is best-first search, where there is a priority queue $Q$ of partial proofs, ordered by some value function $v(\cdot)$. At step $t$, we pop one partial proof $\bar{y}_{1:t}$ (each associated with its current state $s_t$) with the highest value:

$$\bar{y}_{1:t} = \arg \max_{\bar{y}_{1:t}'} v(q \circ \bar{y}_{1:t}'). \quad (2)$$

We then expand $\bar{y}_{1:t}$ by generating $M$ candidate proofsteps to construct the candidate set; where each candidate partial proof $\bar{y}_{1:t+1} \in S_{t+1}(\bar{y}_{1:t})$ is inserted into the queue $Q_{\bar{y}}$ prioritized by the value:

$$Q_{\bar{y}} \leftarrow Q_{\bar{y}} \cup \{(\bar{y}_{1:t+1}, v(q \circ \bar{y}_{1:t+1})) \mid \bar{y}_{1:t+1} \in S_{t+1}(\bar{y}_{1:t})\}.$$ 

The search continues until a full proof $\bar{y}_{1:T}$ is found by equation 2, or termination criteria is reached.

---

### Glossary for theorem proving and reasoning that we use throughout the paper.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Theorem statement</strong></td>
<td>A mathematical statement</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>A statement in the context of a proofsearch, denoted as $s$.</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>A representation containing the contexts and goals for the proof; for simplicity, we use this term interchangeably with goal.</td>
</tr>
<tr>
<td><strong>Proofstep / tactic</strong></td>
<td>A reasoning step that uses established assumptions etc to achieve the goal.</td>
</tr>
<tr>
<td><strong>Planning</strong></td>
<td>A sub-type of reasoning on deriving high-level goals that trigger low-level steps.</td>
</tr>
<tr>
<td><strong>Low-level search</strong></td>
<td>The sampling and pruning for proofsteps.</td>
</tr>
<tr>
<td><strong>High-level search</strong></td>
<td>The sampling and pruning for goals, see Section 3 for details.</td>
</tr>
</tbody>
</table>
3. Method

In this section, we discuss how our method learns and approximates the optimal policy $\pi^*(y_{t+1} \mid s_t)$. In general, the idea of RiR is like having high-level strategies before executing low-level tactics. We employ a planner to set the strategy, and conditional on the strategies, the actor will commit the tactics, while tree search happens in both stages.

3.1. Offline Learning Stage: Goal-Driven Co-Training

Unlike classical approaches which directly learn to minimize the cross-entropy loss with regard to the conditional distribution as shown in equation 1, we propose to learn the joint distribution $p^*(s^*_{t+1}, y^*_{t+1} \mid s_t)$, where $s^*_{t+1}$ is the target goal state achieved by applying $y^*_{t+1}$. Our strategy is very simple: we co-train a goal predictor model $p(s^* \mid s)$ and a goal-driven reasoning step predictor model $p(y^* \mid s, s^*)$, with the following co-training loss:

$$
\mathcal{L}_{\text{col}}(\theta) = -\frac{1}{N} \sum_{(s, y^*, s^*) \sim D} \log p_{\theta}(s^* \mid s) + \log p_{\theta}(y^* \mid s, s^*)
$$

(3)

We use the following input-output prompt format in training for the theorem proving task:

**Planner** (Target Goal Generation):

> **Input:** [CURRENT GOAL] {CURRENT_GOAL s}
> **Output:** {TARGET_GOAL s^*}

**Actor** (Goal-Driven Tactic Generation):

> **Input:** [CURRENT GOAL] {CURRENT_GOAL s}
> **Output:** {TACTIC y^*}

The optimal policy $\pi^*(y_t \mid s_t)$ can be obtained by marginalizing over the predicted target goal states:

$$
\pi^*(y_t \mid s_t) = \int p(s^*_{t+1} \mid s_t) \cdot p(y_t \mid s_t, s^*_{t+1}) \, ds^*_{t+1}
$$

(4)

By decomposing the decision making process into goal state generation and goal-driven reasoning steps generation, and learning through co-training, RiR naturally captures the hierarchical structure of the reasoning.

Notably, using the co-training objective per se ensures a better generalization guarantee on learning the true distribution $p^*(y^* \mid s)$, we present the first information-theoretic analysis in Section 4.1 with the Co-Training Advantage Theorem.

3.2. Online Planning Stage: Goal-Driven Hierarchical Search

We now explain how RiR works via sampling and pruning through hierarchical tree, and present practical implementation choices.

3.2.1. The Modular Design of RiR

Algorithm 1 is a general design for RiR during the online planning phase. The high-level search explores promising target goals, while the low-level search finds the promising tactics to achieve each target goal. A key feature then is the joint update of both trees, based on the the low-level feedback / verification, enabling a unified optimization for reasoning. This hierarchical decomposition enables efficient navigation of the search space by prioritizing the most relevant subgoals at each level. The search processes are guided by flexible policies, allowing the plug-in of various search algorithms (e.g., best-first search and Monte Carlo tree search) and domain-specific heuristics.

**Algorithm 1** RiR – A General Framework

**Input:** problem statement $q$, a language model with parameter $\theta$

1: $\text{tree} \leftarrow \text{Tree}(\theta, q)$
2: repeat
3: $\{s_i\} \leftarrow \text{tree}.\text{policy}()$ \quad \triangleright high-level search
4: $\text{tree} \leftarrow \text{Tree}(\theta, \{s_i\})$
5: repeat
6: $\{y_i\} \leftarrow \text{tree}.\text{policy}()$ \quad \triangleright low-level search
7: until STOP_LOW
8: $\{\text{tree}, \text{tree}\}.\text{update}()$ \quad \triangleright joint update
9: until STOP_HIGH
10: return $\text{tree}.\text{solution}$

3.2.2. Practical Implementation with Best-First Search

In Algorithm 2, we present a best-first search version of RiR, which maintains a priority queue of trajectories, where the priority of a trajectory is determined by its joint negative log-likelihood, defined as:

$$
-\log p(\tau) = -\sum_{i=1}^t \log p(y_{i+1}, s^*_{i+1} | s_i)
$$

$$
= -\sum_{i=1}^t \left( \log p(s^*_i | s_{i-1}) + \log p(y_{i+1}, s^*_{i+1}, s_i) \right).
$$

At each iteration, the algorithm pops the highest-priority trajectory. By prioritizing trajectories with goal-conditioned policy heuristics, RiR efficiently explores the most promising reasoning paths in a more compact, information-directed search space. We present deeper theoretical insights on the efficiency bonus in Section 4.2 by the information gain view with the Hierarchical Planning Advantage Theorem.
Algorithm 2 RiR – Best-First Search

Input: problem statement $q$, a language model with parameter $\theta$

1: $Q \leftarrow$ Queue($q$)
2: while $Q \neq \emptyset$ and not BudgetExhausted() do
3: \[ \tau = (s_0, (s_1, y_1), \ldots, (s_t, y_t), s_t) \leftarrow Q.Pop() \]
4: if $s_t$ is ProofFinished then return $\tau$
5: end if
6: \[ \forall \gamma \text{ high-level search } \] / \[ \forall \gamma \text{ low-level search } \]
7: $G_t \leftarrow$ SampleTargetGoals($s_t, \theta$)
8: for $\hat{s}_{t+1}^{(0)} \in G_t$ do
9: $\forall \gamma \text{ joint update }$
10: $Y_t^{(0)} \leftarrow$ SampleTactics($\hat{s}_{t+1}^{(0)}, s_t, \theta$)
11: $\forall \gamma \text{ variable can help in predicting }$
12: $Y_t^{(0)} \leftarrow$ SampleTactics($\hat{s}_{t+1}^{(0)}, s_t, \theta$)
13: $s_{t+1}^{(0)} \leftarrow$ ApplyTactic($y_t^{(0)}, s_t$)
14: $\forall \gamma \text{ is informative about the }$
15: $\tau' \leftarrow (s_0, \ldots, s_t, (\hat{s}_{t+1}^{(0)}, y_{t+1}), \hat{s}_{t+1}^{(0)})$
16: end for
17: end for
18: end while
19: return Failure

4. Theory: An Information Gain Perspective

The simple insight is that the new mechanism of RiR increases information learned from environments, improving both generalization for reasoning step learning and exploration for reasoning path planning.

4.1. Generalization Guarantee for Goal-Driven Policy Co-Training

In this section, we show that the policy learned through the co-training objective can approximate the true distribution $p^*(y^*|s)$ better than the classical objective, with a non-trivial margin relevant to the conditional information gain provided by predicting the target goal.

Assumption 1 The conditional mutual information between the optimal action $y^*$ and the optimal target goal $s^*$, given the current state $s$, is bounded by a constant $\gamma_I > 0$:

\[ I(y^*; s^*|s) \geq \gamma_I. \tag{5} \]

Assumption 2 Let $p^*(s, y^*)$ be the true joint distribution over triplets $\{(s_i, y_i, s_i')\}_{i=1}^N$. Let $p_\theta(y^*|s)$ and $p_{\theta_{co}}(y^*|s)$ be the learned distributions for the classical and the co-training approach from minimizing the empirical loss $L_c(\theta)$ in Eq. 1 and $L_{co}(\theta)$ in Eq. 3. We assume:

1. The hypothesis classes $\Theta_c$ and $\Theta_{co}$ have VC dimen-

sions $d_c$ and $d_{co}$ and are such that:

\[ L_c(\theta^*_c) \leq \inf_{\theta \in \Theta_c} L_c(\theta) + O \left( \frac{d_c + \log(1/\delta)}{N} \right), \]

\[ L_{co}(\theta^*_{co}) \leq \inf_{\theta \in \Theta_{co}} L_{co}(\theta) + O \left( \frac{d_{co} + \log(1/\delta)}{N} \right), \]

with probability at least $1 - \delta$ over the choice of the training set, where $\theta^*_c = \arg \min_{\theta \in \Theta_c} L_c(\theta)$ and $\theta^*_{co} = \arg \min_{\theta \in \Theta_{co}} L_{co}(\theta)$.

2. The number of training examples $N$ is sufficiently large such that $N \geq \frac{32(d_{co} + \log(1/\delta))}{\gamma_I^2}$.

While Assumption 2 is standard in learning theory [Shalev-Shwartz and Ben-David, 2014], Assumption 1 offers a fresh information-theoretic perspective. The simple intuition is that as long as the target state $s^*$ is informative about the reasoning step $y^*$, learning to predict $s^*$ as an intermediate variable can help in predicting $y^*$, given the current state.

Lemma 1 (Loss Decomposition with Information Gain) Let $L_{co}(\theta^*_{co})$ be the optimal co-training loss and $L_c(\theta^*_c)$ be the optimal classical loss. Suppose the conditional mutual information satisfies $I(y^*; s^* | s) \geq \gamma_I$ for some constant $\gamma_I > 0$. Then, there exists a constant $C > 0$ such that:

\[ L_{co}(\theta^*_{co}) \leq L_c(\theta^*_c) + C\gamma_I. \]

This provides an upper bound on the optimal co-training loss in terms of the optimal classical loss and the conditional mutual information, that leads to our main theorem.

Theorem 1 (Co-Training Advantage) By Assumption 1 and 2, with probability at least $1 - 2\delta$ over the choice of the training set, the following inequality holds:

\[ \mathbb{E}_{p^*(s)} \| ||p^*(y^* | s) - p_{\theta_c}(y^* | s)||_{TV} \| \geq \mathbb{E}_{p^*(s)} \| ||p^*(y^* | s) - p_{\theta_{co}}(y^* | s)||_{TV} \| + \frac{1}{4} \sqrt{\frac{\gamma_I}{2}}. \]

Proof sketch. We express the above total variation distance in the form of KL divergence using Pinsker’s inequality, and show that the latter is smaller by mutual information decomposition and various inequality relationship like triangle inequality and data processing inequality, etc. Detailed proofs for the above results are in the Appendix A.

Remarks. We thus has provided the first formal guarantee on co-training with (current_goal, tactic, target_goal) triplets, showing that this can provably lead to better generalization on the optimal policy distribution, and the benefit comes from the information gain obtained by predicting future target goals.
4.2. Efficiency Guarantee for Goal-Driven Hierarchical Planning

In our hierarchical planning approach for theorem proving, we introduce a target goal space $\mathcal{S} = S$. At step $t$, given the current state $s_t$, we first search for target goals given the current goal; next, conditional on the chosen target goals, we search for tactics, and apply the tactics to transit to new states; the process repeats until the termination (e.g., theorem gets proved). In contrast, the classical single-level planning approach only samples low-level tactics continuously without any high-level guidance; we refer to this as the flat planning, since the planning takes place only in the flat search space. To sum up:

- **(Classical) Flat planning:** we have a policy $\pi_f : S \to A$ that maps states to actions.

- **(RiR) Hierarchical planning:** we have:
  - A high-level planner policy $\pi_h : S \to \mathcal{S}$, that maps current goals to target goals.
  - A low-level actor policy $\pi_l : S \times \mathcal{S} \to A$, that maps current goals and target goals to actions.

**Assumption 3** Assume that the optimal policies $\pi^*_f, \pi^*_h, \pi^*_l$ exist, and let $\hat{\pi}_h$ and $\hat{\pi}_l$ be the learned high-level and low-level policies, we assume approximation errors below:

- **The high-level planner policy:**
  \[
  \mathbb{E}_{s \sim \rho_h^*} [D_{KL}(\hat{\pi}_h^*(\cdot|s)||\pi_h^*(\cdot|s))] \leq \epsilon_h.
  \]

- **The low-level actor policy:**
  \[
  \mathbb{E}_{(s, s^*) \sim \rho_l^* \pi_l^*} [D_{KL}(\hat{\pi}_l^*(\cdot|s, s^*)||\pi_l^*(\cdot|s, s^*))] \leq \epsilon_l.
  \]

where $\rho_h^*$ and $\rho_l^* \pi_l^*$ are the stationary distributions over states and state-subgoal pairs induced by the optimal policies, respectively, and $\epsilon_h, \epsilon_l > 0$.

**Theorem 2 (Hierarchical Planning Advantage)**

Consider a hierarchical planning approach with a high-level policy $\pi_h$ and a low-level policy $\pi_l$, and a flat planning approach with a policy $\pi_f$. Let $N_h(\epsilon)$ and $N_f(\epsilon)$ be the number of node expansions required by the hierarchical and flat planning approaches to find an $\epsilon$-optimal solution w.r.t. at least $1 - \delta$. Under Assumptions 1 and 3, there exist constants $c_1, c_2 > 0$ such that:

\[
\mathbb{E} [N_h(\epsilon)] \leq c_1 e^{-\gamma_l} \cdot \log \left( \frac{1}{\delta} \right) \cdot \mathbb{E} [N_f(\epsilon + c_2 \max(\epsilon_h, \epsilon_l))] \tag{6}
\]

where $\gamma_l$ is the conditional mutual information between the optimal action and the optimal target goal, and $\epsilon_h$ and $\epsilon_l$ are the $\epsilon$-optimality gaps of the learned high-level and low-level policies, respectively.

Detailed proofs are presented in Appendix C.

**Remarks.** This suggests that hierarchical planning can be significantly more sample-efficient when:

- The target goals are informative of optimal actions (high $\gamma_l$), implying that the constrained actions spaces are smaller than the raw spaces (low $H(y^*|s, s^*)$).
- The approximation errors are small compared to the information gain (small $\epsilon_h$ and $\epsilon_l$).

In essence, RiR is helpful when target goals effectively decompose the problem into smaller subproblems while preserving the essential information about the optimal solution. Intuitively, if the target goals selected by the high-level policy provide useful information towards the optimal actions, the low-level policy can focus on a smaller set of relevant actions, leading to more efficient search. Our new RiR framework presents a unified mechanism for this purpose via goal-driven learning and information-directed planning.

5. Experiments

5.1. Settings

**Offline training datasets.** We use LeanDojo Benchmark 4 [Yang et al., 2023] to train language model reasoners. The benchmark is one of the largest formal theorem proving datasets in Lean4, with approximately 100K theorems/proofs and 250K proofsteps with states, extracted from mathlib4\(^4\). We use the training partition in the random split, i.e., around 300M characters for training.

**Environment.** We use the gym-like environment [Brockman et al., 2016] from LeanDojo for neural theorem proving. Given a theorem statement, the task is to generate a formal proof that is verified by Lean. In this environment, language model reasoner can observe the proof state $s$, sequentially run tactics/proofsteps $y$ to transit through states, and receive feedback $r$ upon errors or proof completion.

**Models.** We consider using BYT5-0.3B [Xue et al., 2021] as our base model, which is a pretrained byte-level encoder-decoder Transformer model based on the T5 architecture, and was adopted in [Yang et al., 2023] with the state-of-the-art performance in theorem proving. We refer to this trained checkpoint of Reprover (w/o retrieval) as our baseline.

**Training settings.** For the BYT5-0.3B model, the default number of training steps is 500K, with learning rate to be $5.0 \times 10^{-4}$ and batch size to be 8. The training takes 12 hours on 8 NVIDIA H100 GPU with 80GB memory, and the evaluation takes approximately 4.5 hours for every 100 theorems per H100 GPU with 4 workers.

\(^4\)github.com/leanprover-community/mathlib4.
Reasoning in Reasoning

Evaluation settings. Besides LeanDojo Benchmark 4, we also consider miniF2F-Lean4 [Zheng et al., 2021] for evaluation, which consists of 488 formalized statements from math competitions and college coursework. We use the Pass@1 metric, i.e., given a to-prove theorem statement, the model can only attempt once to find the proof. For low-level proofstep generation, the search width is set to be 64; for high-level goal generation of RiR, the search width is 5. We set a 10-min timeout constraint for each proof.

5.2. Main Results

In this section, we provide brief discussion on the current experiments which shows the state-of-the-art proving performance as well as efficiency of RiR.

Table 2. Performance. Pass@1 rate on LeanDojo and miniF2F.

<table>
<thead>
<tr>
<th>Dataset (→)</th>
<th>miniF2F-test⁶</th>
<th>LeanDojo-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method (→)</td>
<td>Model (→)</td>
<td></td>
</tr>
<tr>
<td>Reprover (BFS)</td>
<td>ByT5-0.3B</td>
<td>ByT5-0.3B</td>
</tr>
<tr>
<td>RiR (BFS)</td>
<td>36.89%</td>
<td>53.73%</td>
</tr>
</tbody>
</table>

Performance Gain. We present the performance comparison of RiR with existing baselines in Table 2. RiR also proved 1 more AIME and 2 more AMC problems compared to the current state-of-the-art Reprover [Yang et al., 2023].

Efficiency Gain. RiR is significantly faster in searching for the optimal reasoning trajectories via a more compact and information-directed search space with the goal-driven planner, as illustrated in Figure 2 on miniF2F benchmark. RiR is more time efficient in the sense that we achieve better results in small computational budget. Specifically, as shown in Figure 3, while the classical Reprover has an average actor time (i.e., time spent for low-level proofstep search) of 78.21s, RiR reduces this to only 23.39s, with additional 3.93s for planner time (i.e., time spent for high-level goal search) on average, setting the new efficiency benchmark for neural theorem proving. In addition, we consider studying the optimal trade-off between the planner budget and the actor budget with policy rollout and lookahead as an interesting future work.

Discussions. We present more example logs showing how RiR found hard proofs fast while classical approaches fail in Appendix D. Take Finset.union_subset_left for example, while the classical method expanded more than 8914 nodes yet still failed after 10 minutes, RiR proved the theorem within 5 seconds and only searched 1 node. We believe the significant improvement in efficiency and effectiveness comes from RiR’s ability to generalize better and to explore better in the more compact and informative search space, empirically supporting our Theorem 1 and 2.

6. Related Works

Reasoning with language models. In the context of language modeling, reasoning typically refers to generating intermediate steps within the language space to reach a final solution to a problem [Wei et al., 2022]. Solving complex...
or novel reasoning problems remains to be an open challenge. One promising direction is reasoning by searching, e.g., expanding the reasoning space by tree search for intermediate steps [Yao et al., 2024; Feng et al., 2023; Liu et al., 2023a; Yuan et al., 2024]. Another research direction is reasoning by decomposition, i.e., generating higher-level goals that trigger a single or a sequence of lower-level steps [Zhou et al., 2022; Liu et al., 2023b; Zheng et al., 2023; Liang et al., 2024; Dalal et al., 2024; Huang et al., 2022; Hu et al., 2024]. The most similar line of literature to ours is subgoal search [Wilkins, 1980; Czechowski et al., 2021; Zawalski et al., 2022; Parascandolo et al., 2020; Paul et al., 2019]. RiR first formally unifies search and decomposing in a hierarchical framework for reasoning with large language models, along with an information-theoretic analysis.

**Automatic Theorem Proving with language models.** As a representative reasoning task, automatic theorem proving (ATP) is often characterized as a tree search problem, i.e., constructing a (tactic-based) proof tree and traversing it to find the correct proof [Li et al., 2024]. In the context of language modeling, proofstep generation forms the edges of the proof tree; the common standard in prior works is to generate single proof steps with the input format similar to \{GOAL\} \{PROOFSTEP\}, i.e., conditional on the current goal, generating the next tactic [Polu and Sutskever, 2020; Yang et al., 2023; Azerbayev et al., 2023; Lample et al., 2022]. For the proof search stage, while people have been using simple heuristics like breadth-first search [Bansal et al., 2019], or MCTS-like search guided by learned value functions [Lample et al., 2022; Polu et al., 2022], designing better search algorithms remains an active area [Li et al., 2024]. The key challenge is that the tactic-based proof space is combinatorially large. Distinguished from prior works, RiR is the first to introduce the goal-driven co-training for proofstep generation with a hierarchical search framework with formal generalization and efficiency guarantee.

**Hierarchical and goal-conditioned RL.** Planning and learning is hard when the decision-making space scales up [Bakker et al., 2004]. Hierarchical RL intends to address this issue by learning a hierarchy of policies operating on different levels of abstraction (e.g., subgoals over the state space). This mitigates the scaling issues by improving exploration for the environment [Ghosh et al., 2020; Chinitis et al., 2022; Kumar et al., 2023; Silver et al., 2023; Le et al., 2018]. There is another line of research termed as goal-conditioned RL [Ghosh et al., 2019; Wang et al., 2023; Ghugare et al., 2024], which trains offline RL policy in a supervised manner conditioning on goal or return. Unlike most prior works that rely on a predefined goal structure, we train models to learn to generate goals in the language space, and refine the goal planning via low-level tree search and joint update.

**Information theory for learning.** Learning is all about information compression, such that neural networks can generalize rather than merely memorize. Information theory thus is a useful tool to analyze learning bounds. For generalization guarantees on supervised learning, common methods include the information bottleneck [Shwartz-Ziv and Tishby, 2017] and PAC-Bayes [McAllester, 1998]; for regret guarantees on online learning, [Russo and Van Roy, 2016] provides a principled Bayesian framework. The analysis of RiR draws insights from both lines of works.

### 7. Conclusions

**Conclusions.** We have developed Reasoning in Reasoning (RiR), the first general framework unifying reasoning by search and reasoning by decomposing. In the domain of automated theorem proving, RiR is practically implemented with goal-driven offline pretraining and hierarchical online planning, where reasoning takes place in different semantic levels hierarchically. We support RiR with rigorous information-theoretical analysis, contributing the Co-Training Advantage Theorem and the Hierarchical Planning Advantage Theorem. These theoretical insights, grounded in information theory, shed light on the fundamental approach for reasoning with language models.

**Limitations and future works.** While we have shown the effectiveness of RiR on neural theorem proving benchmarks, there is a lot more to be built upon our framework. Future directions may include: (i) incorporating dedicated reward models in the planning phase (rather than using the likelihood heuristics); (ii) adding post-training during planning using techniques like contrastive preference learning [Hejna et al., 2023], to further tune the model with pairs of successful and failed reasoning trajectories for self-improvement [Hosseini et al., 2024]; (iii) integrating language feedback [Cheng et al., 2023], contextual information [Welleck and Saha, 2023], and other broader goals for co-training and planning; (iv) exploring the RiR’s applicability to more general domains beyond neural theorem proving; (v) investigating in-context learning alternatives for co-training to apply RiR in black-box models; (vi) adding goal rollout and lookahead to further improve RiR’s performance and efficiency.

---

7Note that our implementation differs from prior frameworks like Draft-Sketch-Proof [Jiang et al., 2022; Zhao et al., 2023], which uses a similar term, hierarchical proof system. These frameworks can be considered as a special case of our approach, with two key differences. First, they generate high-level goals all at once, rather than in a step-by-step search manner. Second, they use informal natural language to represent subgoals, whereas we use formal goal states. These two differences allow our framework to introduce precise search space abstraction with guaranteed generalization and efficiency. Nevertheless, our framework is general enough to still implement informal-to-formal settings when needed, and we leave it to future works.
References


Reasoning in Reasoning


A. Proof for Theorem 1: The Co-Training Advantage Theorem

A.1. Bounding the Total Variation Distance for the Classical Loss

Proof: By Pinsker’s inequality [Cover, 1999], we have:

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_c}(y^* | s) \right\|_{TV} \right] \leq \sqrt{\frac{1}{2} \mathbb{E}_{p^*(s)} \left[ D_{KL}(p^*(y^* | s) || p_{\theta_c}(y^* | s)) \right]}.
\]  
(7)

Next, we relate the expected KL divergence to the classical loss \( L_c(\theta) \). By the definition, we have:

\[
\mathbb{E}_{p^*(s)} \left[ D_{KL}(p^*(y^* | s) || p_{\theta_c}(y^* | s)) \right] = \mathbb{E}_{p^*(s,y^*)} \left[ \log \frac{p^*(y^* | s)}{p_{\theta_c}(y^* | s)} \right] = \mathbb{E}_{p^*(s,y^*)} \left[ - \log p_{\theta_c}(y^* | s) - H(y^* | s) \right] = L_c(\theta) - H(y^* | s),
\]  
(8)

where \( H(y^* | s) \) is the conditional entropy of \( y^* \) given \( s \) under the true distribution \( p^* \).

Now, let \( \vartheta^*_c = \arg \min_{\theta \in \Theta_c} L_c(\theta) \) be the optimal parameters. By the assumption on the hypothesis class \( \Theta_c \), we have with probability at least \( 1 - \delta \) over the choice of the training set that:

\[
L_c(\vartheta^*_c) \leq \inf_{\theta \in \Theta_c} L_c(\theta) + O \left( \sqrt{\frac{d_c \log(1/\delta)}{N}} \right),
\]  
(9)

where \( d_c \) is the VC dimension of \( \Theta_c \) and \( N \) is the size of the training set [Shalev-Shwartz and Ben-David, 2014].

Combining equation 7, equation 8, and equation 9, we obtain the following bound on the total variation distance between \( p^*(s) \) and \( p_{\theta_c}(y^* | s) \):

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_c}(y^* | s) \right\|_{TV} \right] \leq \sqrt{\frac{1}{2} \left( L_c(\vartheta^*_c) - H(y^* | s) \right)} \leq \sqrt{\frac{1}{2} \left( \inf_{\theta \in \Theta_c} L_c(\theta) + O \left( \sqrt{\frac{d_c \log(1/\delta)}{N}} \right) - H(y^* | s) \right)}.
\]  
(10)

A.2. Bounding the Total Variation Distance for the Joint Loss

Similarly, by Pinsker’s inequality, we have:

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_{TV} \right] \leq \sqrt{\frac{1}{2} \mathbb{E}_{p^*(s)} \left[ D_{KL}(p^*(y^* | s) || p_{\theta_{co}}(y^* | s)) \right]}.
\]  
(11)

Next, we express the expected KL divergence in terms of the joint distribution \( p^*(y^*, \tilde{s}^* | s) \) and the conditional mutual information \( I(y^*; \tilde{s}^* | s) \) using the chain rule for KL divergence:

\[
\mathbb{E}_{p^*(s)} \left[ D_{KL}(p^*(y^* | s) || p_{\theta_{co}}(y^* | s)) \right] = \mathbb{E}_{p^*(s)} \left[ D_{KL}(p^*(y^*, \tilde{s}^* | s) || p_{\theta_{co}}(y^*, \tilde{s}^* | s)) \right] - I(y^*; \tilde{s}^* | s).
\]  
(12)

Again, we relate the expected KL divergence to the co-training loss \( L_{co}(\theta) \):

\[
\mathbb{E}_{p^*(s,y^*, \tilde{s}^*)} \left[ \log \frac{p^*(y^*, \tilde{s}^* | s)}{p_{\theta_{co}}(y^*, \tilde{s}^* | s)} \right]
\]
Reasoning in Reasoning

\[ d \]

To restate from the above, we have the total distance from

Substituting equation 18 into equation 17, we obtain:

and \( H(y^*; \tilde{s}^* | s) \) is the conditional joint entropy of \( y^* \) and \( \tilde{s}^* \) given \( s \) under the true distribution \( p^* \).

Similarly, let \( \theta_{co}^* = \arg \min_{\theta \in \Theta_{co}} \mathcal{L}_{co}(\theta) \) be the optimal parameters for the co-training approach. By the assumption on the hypothesis class \( \Theta_{co} \), we have with probability at least \( 1 - \delta \) over the choice of the training set:

where \( d_{co} \) is the VC dimension of \( \Theta_{co} \) and \( N \) is the number of training examples.

Now, based on the assumption that \( I(y^*; \tilde{s}^* | s) \geq \gamma_I \) for some constant \( \gamma_I > 0 \), and given equation 11, equation 12, equation 13, and equation 14, we have:

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_{TV} \right] \\
\leq \sqrt{\frac{1}{2} \left( \mathcal{L}_{co}(\theta_{co}^*) - H(y^*; \tilde{s}^* | s) - I(y^*; \tilde{s}^* | s) \right)} \\
\leq \sqrt{\frac{1}{2} \left( \inf_{\theta \in \Theta_{co}} \mathcal{L}_{co}(\theta) + O \left( \sqrt{\frac{d_{co} + \log(1/\delta)}{N}} \right) - H(y^*; \tilde{s}^* | s) - \gamma_I \right)}.
\]

A.3. Combining the Bounds

In this subsection, we combine the bounds derived from the above and show that with mild assumptions, the total variation distance between \( P^*(y^* | s) \) and \( P_{\theta_{co}}(y^* | s) \) is w.h.p smaller than the total variation distance between \( P^*(y^* | s) \) and \( P_{\theta_{co}}(y^* | s) \) by a non-negative margin of at least \( \frac{1}{4} \sqrt{\frac{\gamma I}{2}} \).

To restate from the above, we have the total distance from \( p^*(y^* | s) \) from the learned distribution by the classical loss and the co-training loss as:

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_{TV} \right] \\
\leq \sqrt{\frac{1}{2} \left( \inf_{\theta \in \Theta_{co}} \mathcal{L}_{co}(\theta) + O \left( \sqrt{\frac{d_{co} + \log(1/\delta)}{N}} \right) - H(y^* | s) \right)}.
\]

By the assumption on \( \mathcal{L}_{co}(\theta_{co}^*) \) and \( \mathcal{L}_{c}(\theta_{co}^*) \), we have:

\[
\inf_{\theta \in \Theta_{co}} \mathcal{L}_{co}(\theta) \leq \mathcal{L}_{co}(\theta_{co}^*) + \frac{\gamma_I}{4} \\
\leq \inf_{\theta \in \Theta_{co}} \mathcal{L}_{c}(\theta_{co}) + O \left( \sqrt{\frac{d_{co} + \log(1/\delta)}{N}} \right) + \frac{\gamma_I}{4}.
\]

Substituting equation 18 into equation 17, we obtain:

\[
\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_{TV} \right]
\]
We start by expressing the difference between the optimal joint loss

\[
L_{\text{co}}(\theta^*) - L_c(\theta^*).
\]

Proof: This completes the proof, showing that under the given assumptions, the co-training approach learns a better conditional distribution for the reasoning step \( y^* \) compared to the classical approach, with a margin of at least \( \frac{3}{4} \gamma_I \). Plainly, this means the co-training approach helps predict the correct next reasoning step more accurately conditional on the current state.

\[
\begin{aligned}
&\leq \frac{1}{2} \left( \inf_{\theta \in \Theta_r} L_c(\theta) + O \left( \sqrt{\frac{d_c + \log(1/\delta)}{N}} \right) \right) + O \left( \sqrt{\frac{d_{co} + \log(1/\delta)}{N}} \right) - H(y^*, \hat{s}^* | s) - \frac{3}{4} \gamma_I. \\
\end{aligned}
\]

(19)

To simplify the bound in equation 19, we use the assumption that \( N \geq \frac{32}{\gamma_I^2}(d_{co} + \log(1/\delta)) \):

\[
O \left( \sqrt{\frac{d_{co} + \log(1/\delta)}{N}} \right) \leq O \left( \sqrt{\frac{\gamma_I}{32}} \right) = O (\sqrt{\gamma_I}).
\]

(20)

Substituting equation 20 into equation 19, and given the fact that \( H(y^*, \hat{s}^* | s) \geq H(y^* | s) \) (information never decrease) and arithmetic mean-geometric mean inequality, we obtain:

\[
\begin{aligned}
&\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_\text{TV} \right] \\
&\leq \frac{1}{2} \left( \inf_{\theta \in \Theta_r} L_c(\theta) + O \left( \sqrt{\frac{d_c + \log(1/\delta)}{N}} \right) - H(y^* | s) - \frac{3}{4} \gamma_I \right) + O (\sqrt{\gamma_I}).
\end{aligned}
\]

(21)

Comparing equation 16 for the classical loss and equation 21 for the co-training loss, we can see that the bound for the co-training approach has an additional term \(-\frac{3}{4} \gamma_I + O (\sqrt{\gamma_I})\). Since \( \gamma_I > 0 \), we have:

\[
\frac{-3}{4} \gamma_I + O (\sqrt{\gamma_I}) \leq -\frac{\gamma_I}{2},
\]

for sufficiently small \( \gamma_I \). Substituting this into equation 21, we obtain:

\[
\begin{aligned}
&\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_\text{TV} \right] \\
&\leq \frac{1}{2} \left( \inf_{\theta \in \Theta_r} L_c(\theta) + O \left( \sqrt{\frac{d_c + \log(1/\delta)}{N}} \right) - H(y^* | s) \right) - \frac{1}{2} \sqrt{\frac{\gamma_I}{2}}.
\end{aligned}
\]

(23)

Combining equation 16 and equation 23, we obtain the main result:

\[
\begin{aligned}
&\mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_\text{TV} \right] \\
&\geq \mathbb{E}_{p^*(s)} \left[ \left\| p^*(y^* | s) - p_{\theta_{co}}(y^* | s) \right\|_\text{TV} \right] + \frac{1}{2} \sqrt{\frac{\gamma_I}{2}}.
\end{aligned}
\]

(24)

This completes the proof, showing that under the given assumptions, the co-training approach learns a better conditional distribution for the reasoning step \( y^* \) compared to the classical approach, with a margin of at least \( \frac{1}{2} \sqrt{\gamma_I} \). Plainly, this means the co-training approach helps predict the correct next reasoning step more accurately conditional on the current state.

\[\Box\]

**B. Proof for Lemma 4.1**

Notice we have the conjecture that the optimal co-training loss \( L_{co}(\theta_{co}^*) \) and the optimal classical loss \( L_c(\theta_c^*) \) satisfy \( L_{co}(\theta_{co}^*) \leq L_c(\theta_c^*) + \frac{2}{4} \). We hereby provide a formal justification to this conjecture.

Proof: We start by expressing the difference between the optimal joint loss \( L_{co}(\theta_{co}^*) \) and the optimal classical loss \( L_c(\theta_c^*) \) in terms of KL divergences. By definition, we have:

\[
\begin{aligned}
L_{co}(\theta_{co}^*) - L_c(\theta_c^*) \\
&= -\frac{1}{N} \sum_{(s, y^*, \hat{s}^*) \in \mathcal{D}^{\text{train}}} \left[ \log p_{\theta_{co}}(\hat{s}^* | s) + \log p_{\theta_{co}}(y^* | s, \hat{s}^*) \right] + \frac{1}{N} \sum_{(s, y^*) \in \mathcal{D}^{\text{train}}} \log p_{\theta_c}(y^* | s).
\end{aligned}
\]
Since KL divergence is non-negative, we have:

$$\mathcal{I}(y^*;\hat{s}^* | s) \geq \gamma_I \quad (i.e., \text{the amount of additional information that the target state provides about the reasoning step given the current state; the insight behind this is that when the target state \(\hat{s}^*\) is informative about the reasoning step \(y^*\), learning to predict \(\hat{s}^*\) can indirectly help in predicting \(y^*\)), we thus can leverage this to bound the second term of equation 28.

Specifically, we have the following fact:

$$\mathcal{I}(y^*;\hat{s}^* | s) = \mathbb{E}_{p^r(s)} \left[ D_{KL} \left( p^*(y^* | s, \hat{s}^*) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right]$$

where the first equality is the definition of mutual information, and the second inequality comes from the convexity of the KL divergence. Essentially, this means that \(\mathcal{I}(y^*;\hat{s}^* | s)\) is lower-bounded by the expected KL divergence between the true conditional distribution \(p^*(y^* | s, \hat{s}^*)\) and the learned conditional distribution \(\bar{p}_{\hat{s}^*}(y^* | s)\) from the classical approach.

Combining equation 29 with the assumption that \(\mathcal{I}(y^*;\hat{s}^* | s) \geq \gamma_I\), we obtain:

$$\mathbb{E}_{p^r(s,\hat{s}^*)} \left[ D_{KL} \left( p^*(y^* | s, \hat{s}^*) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] \geq \gamma_I. \tag{30}$$

By the triangle inequality of KL divergence, we can relate the terms in equation 28 and equation 30 as:

$$\mathbb{E}_{p^r(s,\hat{s}^*)} \left[ D_{KL} \left( p^*(y^* | s, \hat{s}^*) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] \leq \mathbb{E}_{p^r(s,\hat{s}^*)} \left[ D_{KL} \left( p^*(y^* | s, \hat{s}^*) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] + \mathbb{E}_{p^r(s)} \left[ D_{KL} \left( p^*(y^* | s) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right]. \tag{31}$$

Substituting equation 30 and equation 31 into equation 28, we obtain:

$$\mathcal{L}_{co}(\theta_{co}^*) - \mathcal{L}_c(\theta_c^*) \leq \mathbb{E}_{p^r(s)} \left[ D_{KL} \left( p^*(y^* | s) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] - \mathbb{E}_{p^r(s,\hat{s}^*)} \left[ D_{KL} \left( p^*(y^* | s, \hat{s}^*) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] \leq \mathbb{E}_{p^r(s)} \left[ D_{KL} \left( p^*(y^* | s) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right] - (\gamma_I - \mathbb{E}_{p^r(s)} \left[ D_{KL} \left( p^*(y^* | s) || \bar{p}_{\hat{s}^*}(y^* | s) \right) \right])$$
\[ E[

\text{Reasoning in Reasoning}
\]

\[ = 2E_{p^*}(s) \left[ D_{KL}(p^*(y^* | s)||p_{\hat{\theta}^*}(y^* | s)) \right] - \gamma_I \leq 2L_c(\hat{\theta}^*_c) - \gamma_I, \tag{32} \]

where the last inequality follows from the fact that the KL divergence upper-bounds the cross-entropy loss [Cover, 1999]. Rearranging equation 32, we obtain:

\[ L_{co}(\hat{\theta}^*_c) \leq L_c(\hat{\theta}^*_c) + \frac{\gamma_I}{2}, \tag{33} \]

which completes our proof. \qed

\section{C. Proof for Theorem 2}

\textbf{Proof:} To start with, let \( T : S \times A \rightarrow S \) be the transition function, where \( T(s, a) \) denotes the next state after taking action \( a \) in state \( s \). Let \( R : S \times A \rightarrow \mathbb{R} \) be the reward function, where \( R(s, a) \) denotes the reward obtained by taking action \( a \) in state \( s \). Define the value function \( V^\pi(s) \) as the expected cumulative reward starting from state \( s \) and following policy \( \pi \):

\[ V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, \pi \right] , \tag{34} \]

where \( \gamma \in (0, 1) \) is the discount factor. The optimal value function is defined as \( V^*(s) = \max_{\pi} V^\pi(s) \), and the optimal policy \( \pi^* \) satisfies \( V^{\pi^*}(s) = V^*(s) \) for all \( s \in S \).

Assuming that there exists a constant \( c > 0 \) such that for any state \( s \in S \), target goal \( s^* \in \hat{S} \), and action \( y^* \in A_{s,s^*} \) such that \( p(y^* | s) \leq \frac{1}{|A_{s,s^*}|} \), we have:

\[ I(y^*:s^*|s) = E_{s,s^*} \left[ \frac{p(y^*|s,s^*)}{p(y^*|s)} \right] \leq E_{s,s^*} \left[ D_{KL}(p(y^*|s,s^*)||p(y^*|s)) \right] \geq E_{s,s^*} \left[ \log \frac{1}{|A_{s,s^*}|} \right] + \log \frac{1}{c}, \tag{37} \]

where \( A_{s,s^*} = \{ a \in A : T(s, a) = s^* \} \) is the set of actions that transition from state \( s \) to target goal \( s^* \). Given \( I(y^*:s^*|s) \geq \gamma_I \), we have:

\[ E_{s,s^*} [|A_{s,s^*}|] \leq \frac{1}{c} e^{-\gamma_I}. \tag{38} \]

This shows that the effective action space in the hierarchical planning approach is reduced by a factor of at least \( e^{\gamma_I} \) in expectation compared to the flat planning approach, multiplied by a scalar.

To bound the expected search complexity of the hierarchical planning approach, we will use the notion of \( \epsilon \)-optimality. A policy \( \pi \) is said to be \( \epsilon \)-optimal if \( V^\pi(s) \geq V^*(s) - \epsilon \) for all \( s \in S \), and we assume that the learned policies \( \hat{\pi}_h, \hat{\pi}_l, \hat{\pi}_f \) are \( \epsilon \)-optimal with respect to their corresponding optimal policies.

To find an \( \epsilon \)-optimal solution, the hierarchical planning approach needs to expand nodes until the expected cumulative reward of the best solution found so far is within \( \epsilon \) of the optimal value function. Let \( \hat{V}(s, s^*) \) be the value function of the best solution found by the hierarchical planning approach for state \( s \) and target goal \( s^* \). The search can be terminated when:

\[ E_{s,s^*}[\hat{V}(s, s^*)] \geq E_{s,s^*}[V^*(s, s^*)] - \epsilon. \tag{39} \]

The hierarchical planning approach needs to expand nodes until:

\[ E_{s,s^*}[\hat{V}(s, s^*)] \geq E_{s,s^*}[V^*(s, s^*)] - (\epsilon_h + \epsilon_l + \epsilon). \tag{40} \]

For the flat planning approach, the termination condition is:

\[ E_s[\hat{V}(s)] \geq E_s[V^*(s)] - (\epsilon_f + \epsilon), \tag{41} \]

\textsuperscript{8}This upper bound assumption may be strong and we intend to improve it in the future work.
where \( \hat{V}(s) \) denotes the value function of the best solution found by the flat planning approach for state \( s \).

Let \( N_h(\epsilon) \) and \( N_f(\epsilon) \) be the number of node expansions required by the hierarchical and flat planning approaches, respectively, to find an \( \epsilon \)-optimal solution. We have:

\[
E[N_h(\epsilon)] \leq E_{s,s^*}[|A_{s,s^*}|] \cdot E[N_f(\epsilon + \epsilon_h + \epsilon_l)]
\]

\[
\leq \frac{1}{c} \cdot e^{-\gamma l} \cdot E[N_f(\epsilon + \epsilon_h + \epsilon_l)].
\]

Applying the Chernoff bound with \( \epsilon = \delta \) and setting the right-hand side to be at most \( \delta \), we get:

\[
N_h(\epsilon) \geq \frac{2}{\delta} \log \left( \frac{1}{\delta} \right).
\]

\[
N_f(\epsilon + \epsilon_h + \epsilon_l) \geq \frac{2}{\delta} \log \left( \frac{1}{\delta} \right).
\]

Combining these bounds, we have:

\[
E[N_h(\epsilon)] \leq \frac{1}{c} \cdot e^{-\gamma l} \cdot E[N_f(\epsilon + \epsilon_h + \epsilon_l)]
\]

\[
\leq \frac{1}{c} \cdot e^{-\gamma l} \cdot \left( \frac{2}{\delta} \log \left( \frac{1}{\delta} \right) + O(1) \right).
\]

To bound the approximation errors \( \epsilon_h \) and \( \epsilon_l \), we assume that the KL divergences between the optimal and learned policies are bounded:

\[
E_{s \sim \rho^{\pi_h}_h}[D_{KL}(\pi_h^*([s])\|\hat{\pi}_h([s]))] \leq \epsilon_h',
\]

\[
E_{(s,s^*) \sim \rho^{\pi_h}_h,\pi^*_h}[D_{KL}(\pi_h^*([s,s^*])\|\hat{\pi}_h([s,s^*]))] \leq \epsilon_l'.
\]

where \( \rho^{\pi_h}_h \) and \( \rho^{\pi_h}_h,\pi^*_h \) denote the stationary distributions over states and state-goal pairs induced by the optimal policies.

By Pinsker’s inequality, we have:

\[
E_{s \sim \rho^{\pi_h}_h}[V^*(s) - V^{\hat{\pi}_h}(s)] \leq \sqrt{\frac{2\epsilon_h'}{1 - \gamma}},
\]

\[
E_{(s,s^*) \sim \rho^{\pi_h}_h,\pi^*_h}[V^*(s,s^*) - V^{\hat{\pi}_h}(s,s^*)] \leq \sqrt{\frac{2\epsilon_l'}{1 - \gamma}}.
\]

By definition of the \( \epsilon \)-optimality and the monotonicity of expectation, we have:

\[
\epsilon_h \leq E_{s \sim \rho^{\pi_h}_h}[V^*(s) - V^{\hat{\pi}_h}(s)] \leq \sqrt{\frac{2\epsilon_h'}{1 - \gamma}},
\]

\[
\epsilon_l \leq E_{(s,s^*) \sim \rho^{\pi_h}_h,\pi^*_h}[V^*(s,s^*) - V^{\hat{\pi}_h}(s,s^*)] \leq \sqrt{\frac{2\epsilon_l'}{1 - \gamma}}.
\]

Thus the approximation errors \( \epsilon_h \) and \( \epsilon_l \) can be bounded in terms of the KL bounds \( \epsilon_h' \) and \( \epsilon_l' \), and the discount factor \( \gamma \).

Combining the bounds from the above, we have:

\[
E[N_h(\epsilon)] \leq \frac{1}{c} \cdot e^{-\gamma l} \cdot \left( \frac{2}{\delta} \log \left( \frac{1}{\delta} \right) + O(1) \right) \cdot E[N_f(\epsilon + \sqrt{\frac{2\epsilon_h'}{1 - \gamma}} + \sqrt{\frac{2\epsilon_l'}{1 - \gamma}})].
\]

Adding simplification with constant terms we obtain:

\[
E[N_h(\epsilon)] \leq c_1 \cdot e^{-\gamma l} \cdot \log \left( \frac{1}{\delta} \right) \cdot E[N_f(\epsilon + c_2 \cdot \max(\epsilon_h, \epsilon_l))].
\]

This completes the proof of the Hierarchical Planning Advantage Theorem. \[\square\]
D. Detailed Experimental Results and Logs

We are open-sourcing all our codes, training scripts, evaluation logs, and trained model checkpoints at this link:

- [https://anonymous.4open.science/r/RiR-6EA1](https://anonymous.4open.science/r/RiR-6EA1).

For evaluation on LeanDojo, we use:

- Commit: fe4454af900584467d21f4fd4fe951d29d9332a7.

For evaluation on miniF2F, we use:

- Commit: 9e445f5435407f014b88b44a98436d50dd7abd00.

We hereby present some example proofs from logs, showing how RiR succeeded with significantly fewer nodes to search. More examples can be found in our released repository.

### Example 0: Proof Found by RiR

**Theorem:**
- File Path: Mathlib/Order/ConditionallyCompleteLattice/Basic.lean
- Full Name: OrderIso.map_ciSup

**Status:** Status.PROVED

**Proof:**
```
simp [iSup, hf]
rw [e.map_csSup']
swap
assumption'
apply Set.range_nonempty
rw [• Set.range_comp]
rfl
```

**Search Statistics:**
- Planner Time: 150.2634212092962
- Actor Time: 315.0649007729953
- Environment Time: 38.92193151102401
- Total Time: 505.9431369260419
- Total Nodes: 2207
- Searched Nodes: 37

### Example 0: Failure by Reprover (w/o retrieval)

**Theorem:**
- File Path: Mathlib/Order/ConditionallyCompleteLattice/Basic.lean
- Full Name: OrderIso.map_ciSup

**Status:** Status.OPEN

**Proof:** None

**Search Statistics:**
- Actor Time: 512.3867035790754
- Environment Time: 89.58101247090963
- Total Time: 602.1384408420126
- Total Nodes: 4082
- Searched Nodes: 160
Example 1: Proof Found by RiR

**Theorem:**
- File Path: Mathlib/Data/Finset/Basic.lean
- Full Name: Finset.union_subset_left

**Status:** Status.PROVED

**Proof:**
- exact Finset.Subset.trans (Finset.subset_union_left s t) h

**Search Statistics:**
- Planner Time: 1.3937767379684374
- Actor Time: 3.304290219093673
- Environment Time: 0.07375576300546527
- Total Time: 4.774586059036665
- Total Nodes: 7
- Searched Nodes: 1

Example 1: Failure by Reprover (w/o retrieval)

**Theorem:**
- File Path: Mathlib/Data/Finset/Basic.lean
- Full Name: Finset.union_subset_left

**Status:** Status.OPEN

**Proof:** None

**Search Statistics:**
- Actor Time: 491.1531239761098
- Environment Time: 110.1171304465338
- Total Time: 601.520013278001
- Total Nodes: 8914
- Searched Nodes: 233
Example 2: Proof Found by RiR

Theorem:
File Path: Mathlib/Data/Nat/PrimeFin.lean
Full Name: Nat.Prime.primeFactors
Status: Status.PROVED
Proof:
  ext
  simp [hp.ne_zero]
  simp [hp, Nat.dvd_prime hp]
  aesop

Search Statistics:
  Planner Time: 150.2634212092962
  Actor Time: 315.0649007729953
  Environment Time: 38.92193151102401
  Total Time: 505.9431369260419
  Total Nodes: 2207
  Searched Nodes: 37

Example 2: Failure by Reprover (w/o retrieval)

Theorem:
File Path: Mathlib/Data/Nat/PrimeFin.lean
Full Name: Nat.Prime.primeFactors
Status: Status.OPEN
Proof: None

Search Statistics:
  Actor Time: 474.4240076234564
  Environment Time: 127.5987611755263
  Total Time: 602.1851601980161
  Total Nodes: 4231
  Searched Nodes: 133
Example 3: Proof Found by RiR

Theorem:
  File Path: Mathlib/Order/SuccPred/Basic.lean
  Full Name: exists_succ_iterate_or

Status: Status.PROVED

Proof:
  obtain h : le_total a b
  exacts [Or.inl (IsSuccArchimedean.exists_succ_iterate_of_le h),
           Or.inr (IsSuccArchimedean.exists_succ_iterate_of_le h)]

Search Statistics:
  Planner Time: 15.921687303110957
  Actor Time: 44.464585242792964
  Environment Time: 8.429574175737798
  Total Time: 68.86368872597814
  Total Nodes: 377
  Searched Nodes: 3

Example 3: Failure by Reprover (w/o retrieval)

Theorem:
  File Path: Mathlib/Order/SuccPred/Basic.lean
  Full Name: exists_succ_iterate_or

Status: Status.OPEN

Proof: None

Search Statistics:
  Actor Time: 519.0408471203409
  Environment Time: 86.30267175737798
  Total Time: 605.4483464460354
  Total Nodes: 2819
  Searched Nodes: 95