PROVABLE BENEFITS OF RLVR OVER SFT FOR REASONING MODELS: LEARNING TO BACKTRACK EFFICIENTLY

Stanley Wei
Princeton University
stanley.wei@princeton.edu

Juno Kim
University of California, Berkeley
junokim@berkeley.edu

ABSTRACT

Recent advances in large language models (LLMs) have demonstrated that reinforcement fine-tuning of pretrained base models can lead to significant gains in reasoning performance at inference time. In this work, we theoretically analyze why reinforcement fine-tuning induces better reasoning ability than purely supervised fine-tuning (SFT) methods. We model chain-of-thought (CoT) reasoning as a pathfinding problem on graphs and compare the popular method of reinforcement learning with verifiable rewards (RLVR) against traditional SFT. We prove that SFT, when trained on golden shortest paths without negative examples, fails to learn how to efficiently backtrack. In contrast, an RLVR-trained model can learn how to efficiently backtrack from dead ends using only outcome reward. This leads to an exponential separation in inference-time compute between the two methods, and demonstrates that RLVR leads the model to learn the location of difficult decisions in a reasoning chain, ultimately allowing for better allocation of inference-time compute. Finally, we show that the reasoning traces of an RLVR model can be distilled to train a base model to backtrack efficiently as well.

1 INTRODUCTION

Modern large language models (LLMs) are trained to achieve strong reasoning capabilities on a wide range of tasks such as mathematical problem-solving and code generation. In the context of LLMs, *reasoning* refers to the generation of long chain-of-thought (CoT) which mimic the step-by-step nature of human reasoning to solve complex logical problems (Wei et al., 2022; Lightman et al., 2023). Reasoning models treat such tasks as a sequential multi-step decision process and deploy strategies utilizing additional test-time compute budget, such as sampling, tree search, aggregation, and backtracking. This approach has proved to yield efficient and scalable gains over initial pretraining (Snell et al., 2024; Muennighoff et al., 2025), and has been adopted with great success in various frontier and open-source models (OpenAI, 2024; Shao et al., 2024; Guo et al., 2025).

In order to learn or strengthen these desired inference-time reasoning behaviors, LLMs must undergo stages of post-training (Kumar et al., 2025; Xu et al., 2025). The post-training procedure typically follows one of two main strategies: supervised fine-tuning (SFT) on expert demonstrations, and reinforcement learning (RL) on feedback from a reward model or task verifier. SFT uses an off-policy dataset of demonstrations created or annotated by an expert (typically a human or a more powerful model) and trains the base model to imitate these responses. While SFT has been the de facto method for post-training, it is prone to memorization or overfitting (Chu et al., 2025a) and can induce uninformative pseudo-reasoning paths (Chen et al., 2025). Moreover, curating high-quality expert demonstrations can be expensive and time-consuming, depending on the nature of the task.

In contrast, methods such as reinforcement learning with verifiable rewards (RLVR) (Wen et al., 2025; DeepSeek-AI et al., 2025a) and reinforcement learning from human feedback (RLHF) (Christiano et al., 2023; Ouyang et al., 2022) learn from reward models or verifiers via on-policy exploration. The RL approach has been argued to generalize more effectively and have the potential to unlock entirely new reasoning capabilities (Chu et al., 2025a; Wang et al., 2025; Zhu et al., 2025). Nevertheless,

we still lack a principled understanding of the differences between the two methods, or a theoretical framework under which to compare various post-training algorithms.

In this paper, we focus on *backtracking ability* as a way to distinguish the effectiveness of post-training methods. Backtracking is a key element of human problem-solving: when a line of approach is revealed to be incorrect or unhelpful, one returns to an earlier decision point and explores an alternative branch. This greatly reduces complexity of the effective search space. Empirically, backtracking has been shown to be greatly beneficial to LLM reasoning ability, either implicitly in the chain of thought (Cai et al., 2025), or explicitly with backtrack tokens (Yang et al., 2025) or rolling back sequence generation (Singh et al., 2025). Hence we are motivated to ask:

Which post-training method teaches the base model to efficiently backtrack at inference time?

Our contributions. We approach this question by modeling CoT reasoning as pathfinding on graphs, a sandbox commonly studied in the literature (Sanford et al., 2024; Bachmann and Nagarajan, 2025; Kim et al., 2025). While existing works have studied the benefits of backtracking from an information or sampling perspective (Shalev-Shwartz and Shashua, 2025; Rohatgi et al., 2025), we provide a novel *dynamical* characterization of running SFT versus RLVR. We design a multigraph similar to the path-star graph (Bachmann and Nagarajan, 2025) with W branches of depth $\sim K$, which the pretrained world model – a linear-softmax bigram over edges, or a trigram over nodes – must learn to navigate by backtracking from failed branches. Our results are summarized as follows.

- We prove that SFT trained only on golden shortest paths does not learn any backtracking strategy, while the RLVR-trained model learns to backtrack consistently using only outcome reward with a length penalty in finite time.
- We show an exponential test-time compute separation: after convergence, the RLVR model can reach any target in $\Theta(WK)$ expected time, while the SFT model requires $\Theta(WL^K)$ time.
- We further show that distilling RLVR-generated reasoning traces to a base model via supervised learning transfers efficient backtracking, recovering $\Theta(WK)$ inference-time compute.

Intuitively, SFT on expert solutions trains the base model to continue along a gold path with no exposure to dead ends. Hence, SFT need not learn an efficient retreat strategy. In contrast, on-policy RLVR necessarily generates and trains on the model's own unsuccessful partial rollouts. This exploration produces gradient signal not only about good forward actions, but also backtracking actions when the model has committed to a poor branch. We formalize this difference via a dynamical analysis of gradient flow (for SFT) and sign policy-gradient flow (for RLVR). Our results provide theoretical support for the importance of backtracking data for reasoning in both RL and distillation.

All proofs are deferred to the appendix.

2 Related Work

Backtracking in LLM reasoning. Backtracking has empirically been used to quantify and improve reliability and efficiency in LLM reasoning. Inference-time search frameworks such as Tree-of-Thoughts explicitly explore and prune a branching space of intermediate thoughts, leading to depth-first or best-first search with backtracking (Yao et al., 2023; Long, 2023), generalized by Graph-of-Thoughts methods (Besta et al., 2024). Qin et al. (2025) study when sequential search with backtracking benefits over parallel sampling. In particular, they empirically show that models with backtracking capabilities benefit greatly from RL finetuning on reasoning tasks, which agree with our theoretical results. Moreover, Singh et al. (2025) propose preemptive backtracking guided by in-context value verification to identify and focus resampling from suspected failure points. Yang et al. (2025) study encouraging models to decide when to revert during reasoning via introducing an explicit backtrack token, aiming to internalize structured backtracking.

From a theoretical perspective, Shalev-Shwartz and Shashua (2025) show the necessity of search and backtracking for certain graph search tasks and parity problems, and describe a learning method which builds a search tree with explicit backtracking. Rohatgi et al. (2025) propose a test-time sampling and backtracking algorithm which uses process rewards, which mixes in time quadratic in the depth of the search tree, allowing for exact sampling from a target distribution on the leaves.

However, these works do not study how to learn such behavior via post-training, which is the focus of our dynamical analysis. Kim et al. (2025) propose a cluster graph model of reasoning and analyze CoT pathfinding as a metastable Markov process. They also provide convergence guarantees for a simple RL method (proximal policy optimization); however, they do not study RLVR or SFT, and do not consider backtracking in CoT.

RL versus SFT. Recent empirical studies have highlighted significant qualitative differences between models post-trained via SFT and those with RL. Chu et al. (2025b) provide a comparative analysis on an arithmetic-based reasoning task that suggests SFT tends to memorize the distribution of training demonstrations, whereas RL facilitates better generalization to out-of-distribution (OOD) tasks. This distinction is further supported by Shenfeld et al. (2025), in which the authors argue that on-policy RL implicitly regularizes the model towards KL-minimal solutions (with respect to the base model), allowing the model to find simpler solutions that are robust to catastrophic forgetting. Park et al. (2025) also report that SFT underperforms RL on the synthetic Countdown task; moreover, RL-only post-training can induce fundamentally improved OOD generalization. In addition, Chen et al. (2025) show that SFT can elicit pseudo-reasoning paths in large vision-language models by only superficially imitating expert models. Such paths often contain uninformative or incorrect reasoning steps, and even hurt subsequent RL training stages. They also propose an RL-based approach which leads to more adaptive reasoning behavior.

Reasoning as graph search. Pathfinding in graphs has been widely used as a sandbox for understanding LLM reasoning capabilities. Abbe et al. (2024) propose the notion of globality degree to capture transformer learning ability, and show that regular transformers cannot efficiently solve cycle tasks. Sanford et al. (2024) study the ability of transformer networks to solve various graph algorithms in terms of their expressivity such as network width and depth. From an empirical perspective, synthetic graph pathfinding tasks have been used to study compositional reasoning ability (Khona et al., 2024) and understand internal prediction mechanisms (Cohen et al., 2025). In particular, the path-star graph (which our construction generalizes) has been shown to be difficult to solve for LLMs, and has been used to study the limitations of reasoners trained via next-token prediction (Bachmann and Nagarajan, 2025; Frydenlund, 2024).

3 Problem Formulation

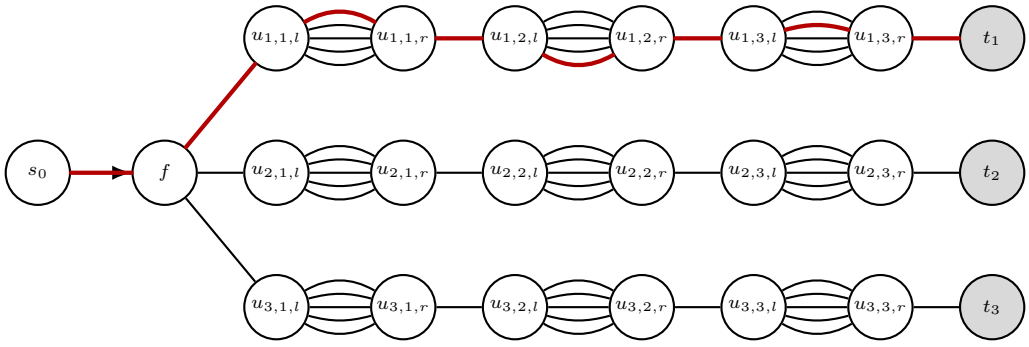

Figure 1: Structure of the world model graph, consisting of a source s_0, a fork f, and W branches. Each branch contains a sequence of K diamonds, each with L multiedges, leading to a final leaf node. Here, $K = W = 3$ and $L = 5$. An example shortest-length path from s_0 to t_1 is highlighted.

3.1 CoT as a pathfinding task

We model our reasoning task as pathfinding on a toy graph that represents the world model (Sanford et al., 2024; Bachmann and Nagarajan, 2025; Kim et al., 2025). Specifically, we consider a multigraph consisting of the following components; see Figure 1.

- Source node s_0, fork node f;
- Diamond $\diamondsuit(u, v)$, which is a subgraph of L undirected multiedges between nodes u, v;

- Leaf nodes t_i.

The topology of the graph is as follows: the source node s_0 is connected to a fork node f through a directed edge to f (this will be the only directed edge in the graph). f now branches out in W directions. In each branch $i = 1, \cdots, W$, we have a sequence of K *diamonds*:

$$\diamondsuit(u_{i,1,l}, u_{i,1,r}), \diamondsuit(u_{i,2,l}, u_{i,2,r}), \ldots, \diamondsuit(u_{i,K,l}, u_{i,K,r})$$

where consecutive left and right diamonds are connected. When multiedges occur, we will denote the edges by $u \overset{j}{\leftrightarrow} v$ for $1 \le j \le L$ for the multiedges in $\diamondsuit(u, v)$. Finally, we connect node $u_{i,K,r}$ to a leaf node t_i. For ease of notation, we will denote $u_{i,0,r} := f$ and $u_{i,K+1,l} := t_i$ for all i.

Reasoning task. In our setup, the reasoning model is prompted with a desired target node u, and the goal is to output a valid path from s to u. We are interested in understanding how post-training a language model with the standard paradigm of supervised fine-tuning (SFT) versus reinforcement learning with verifiable rewards (RLVR) influences the test-time behavior of the model, when prompted with a new reasoning task.

3.2 PRETRAINING

We consider a bigram model, in which each state consists of an edge and traversal direction (*not* the underlying vertices); that is, for an edge $e := u \leftrightarrow v$ in the graph, $u \to v$ and $v \to u$ are valid states. Furthermore, valid state transitions from an edge $u \to v$ would be to states in the set $N_v := \{v \to w : v \leftrightarrow w \in E\}$ (in particular, note that multiedges are distinct). For the model, we encode each state x as a one-hot vector in $\mathbb{R}^{2|E|}$, and we use a single-layer softmax predictor as follows:

$$\pi_\Theta(\cdot|x) = \mathrm{softmax}(\langle \Theta, x \rangle), \quad \Theta \in (\mathbb{R} \cup \{-\infty\})^{2|E| \times 2|E|}.$$

We remark that this model is strictly more expressive than a *trigram model over nodes*: any trigram $\varphi(a|b, c)$ can be encoded as $\pi_\Theta(b \leftrightarrow c|a \leftrightarrow b)$, and moreover φ cannot express multiedges.

Initially, the predictor learns the world model or underlying graph, which we view as the pretrained model; this viewpoint has been used to study pathfinding tasks by Kim et al. (2025). For a given edge $u \to v$, we set the transition probabilities to be $1/|N_v|$ to edges in N_v, and 0 otherwise. When Θ is clear from context, we will often omit the subscript and denote the policy simply as π. We will also denote the underlying distribution of the described world model as $\mathcal{D}$, from which we draw samples (s, a) for current edge state and next edge state pairs such that the marginal probability over s is uniform over all edge states.

Theorem 1 (Warm-up: convergence of pretraining)**.** *Suppose we train on the pairs $(s, a) \sim \mathcal{D}$ of current edge state and next edge state for our bigram such that $\mathbb{P}_{(s,a)\sim\mathcal{D}}[s]$ is uniform for all states. Under the loss*

$$L_{\mathrm{pre}}(\Theta) := \mathbb{E}_{(s,a)\sim\mathcal{D}}[-\log \pi_\Theta(a|s)],$$

standard gradient flow from zero initialization learns transition probabilities in finite time.

The proof of Theorem 1 is deferred to Section B of the appendix. This justifies the following assumption on exactness of pretraining:

Assumption 1. *The pretrained model $\pi_{\Theta_{\mathrm{pre}}}$ has converged arbitrarily close to the world model's distribution.*

This simplification is also used in Kim et al. (2025) to focus on analysis of post-training. Hence, $\pi_{\Theta_{\mathrm{pre}}}$ can also be seen as a trigram over nodes, with multiedges taken into account.

3.3 POST-TRAINING

We study post-training π_Θ with either SFT or RLVR. In both setups, we consider training on data corresponding to path-target pairs, where the targets are sampled from the W leaf nodes t_i. However, the main difference is that SFT is given these pairs, whereas RLVR must obtain them via on-policy rollouts. We detail these methods in the following sections. In both cases, we assume the generation does not depend on the prompted target. That is, it is a pure bigram or Markov chain with initial state $s \to f$ and transition probabilities $\pi_\Theta(\cdot)$.

3.3.1 SFT

For SFT, we assume the model has access to golden shortest-length path examples, with no backtracking or exploration data. In practice, this corresponds to (for instance) feeding the model with gold standard solutions to a math problem. In particular, we train on shortest-length paths from the source s_0 to one of the W leaf nodes t_i for $1 \le i \le W$. These golden paths have the following characteristics: ((1) it chooses the correct branch at the fork node f, and (2) for each diamond $\diamondsuit(u_{i,j,l}, u_{i,j,r})$ for $1 \le j \le K$, it chooses exactly one of the L multiedges to traverse.

We optimize the cross-entropy loss over the dataset of golden paths and targets:

$$\min_{\Theta} L(\Theta) = \mathbb{E}_{x\sim\mathcal{U}(t_i), y\sim\mathcal{D}_x}\left[-\sum_{t=0}^{|y|-1} \log \pi_\Theta(y_{t+1}|y_t)\right] \tag{1}$$

where $\mathcal{D}_x$ is any (fully supported) distribution over golden shortest-length paths from the source to the target x. For ease of exposition, we consider optimizing this loss function via gradient flow, so that:

$$\frac{\mathrm{d}\Theta_{s,a}}{\mathrm{d}t} = -\frac{\partial L}{\partial \Theta_{s,a}}$$

for all pairs s, a of current and next states, respectively.

3.3.2 RLVR

For RLVR, we generate on-policy rollouts starting from the source s_0, with a verifier which checks whether the target node t_i has been reached. As with many open-source models (Kimi et al., 2025; DeepSeek-AI et al., 2025b), we will employ the use of a length penalty along with the verifier outcome reward, which for pathfinding is simply the length of the rollout. That is, our loss function is

$$\begin{aligned} \max_{\Theta} J(\Theta) &= \mathbb{E}_{x\sim\mathcal{U}(t_i), y\sim\pi_\Theta}[r(x,y)], \\ r(x,y) &= \mathbf{1}\{y \text{ hits node } x\} - \beta|y|, \end{aligned} \tag{2}$$

with an appropriately chosen length penalty β. Alternatively, we could choose to use a finite horizon instead of a length penalty and heuristically expect the same results.

For a given rollout with target $x = t_i$, we assume that the verifier stops the rollout as soon as the current state s becomes $u_{i,K,r} \to x$. Then with a slight abuse of notation, $\mathbb{E}_{y\sim\pi}[|y|]$ represents the expected hitting time of t_i (which is same for all targets by symmetry).

Policy gradient update. For the dynamical analysis, we consider the policy gradient update (Sutton et al., 1999):

$$\nabla_\Theta J(\Theta) = \mathbb{E}_x \mathbb{E}_{y\sim\pi}[r(x,y)\nabla_\Theta \log \pi_\Theta(y)] = \mathbb{E}_x \mathbb{E}_{y\sim\pi}\left[r(x,y)\sum_{t=0}^{|y|-1} \nabla_\Theta \log \pi_\Theta(s_{t+1}|s_t)\right]$$

where s_t is the current edge state, s_{t+1} is a potential next edge state, and the policy does not depend on the target.

Without loss of generality, we will work with the case where the length penalty strength $\beta = 1$. For simplicity of analysis, we consider policy gradient optimization via signed gradient flow. That is,

$$\frac{\mathrm{d}\Theta_{s,a}}{\mathrm{d}t} = \mathrm{sgn}\left(\frac{\partial J}{\partial \Theta_{s,a}}\right)$$

for all pairs s, a of current and next state, respectively. Signed gradient descent has also been studied in Kim et al. (2025) to avoid fringe non-convergence issues.

4 Main Results

In this section, we present our main results, with proof sketches deferred to Section A of the appendix and full proofs in the remainder of the appendix.

4.1 NOTATION

To rigorously analyze the two algorithms, we first observe that by symmetry of the population gradient and the graph topology, the transition probabilities for "topologically equivalent" edge states on different branches (i.e., at the same depth and same orientation with respect to the fork) are identical. Thus for any given j, the transition probabilities at any time $t \geq 0$ are the same for the following state types:

1. Forward (resp. backward) diamond connector states $u_{i,j-1,r} \to u_{i,j,l}$ (resp. $u_{i,j,l} \to u_{i,j-1,r}$) for all $1 \leq i \leq W$.
2. Forward (resp. backward) diamond multiedge states $u_{i,j,l} \overset{(\ell)}{\to} u_{i,j,r}$ (resp. $u_{i,j,r} \overset{(\ell)}{\to} u_{i,j,l}$) for all $1 \leq i \leq W$ and $1 \leq \ell \leq L$.

Due to multiedge states being identical logit-wise regardless of the choice of target, we will treat those states as identical; transitions into those states will have equal probability by symmetry. Hence, we denote a single multiedge state $u_{i,j,l} \to u_{i,j,r}$ to be the aggregate of $u_{i,j,l} \overset{(\ell)}{\to} u_{i,j,r}$, and similarly for $u_{i,j,r} \to u_{i,j,l}$.

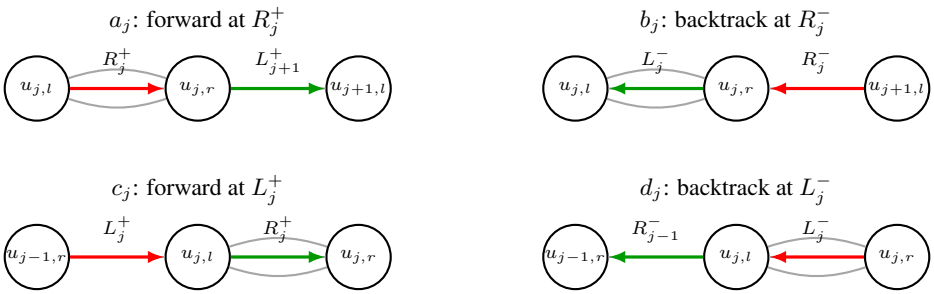

Figure 2: Desired edge-state transition types a_j, b_j, c_j, d_j. The current edge-state is denoted by **red**; the desired next edge-state is denoted by **green**.

We also use the following notation for states; see Figure 2.

1. $R_{i,j}^+$: state $u_{i,j,l} \to u_{i,j,r}$ (arrived at the right diamond node from the left). The probability of the forward connector (same direction as edge state) is denoted a_j, and the reverse multiedges back across the diamond have total probability $1 - a_j$ ($\frac{1-a_j}{L}$ per edge).
2. $R_{i,j}^-$: state $u_{i,j+1,l} \to u_{i,j,r}$ (arrived at the right diamond node from the right). The total probability of the backward multiedges is b_j ($\frac{b_j}{L}$ per edge), and the state back across the same edge has probability $1 - b_j$.
3. $L_{i,j}^+$: state $u_{i,j-1,r} \to u_{i,j,l}$ (arrived at the left diamond node from the left). The total probability of the forward multiedges is c_j, and the state back across the same edge has probability $1 - c_j$.
4. $L_{i,j}^-$: state $u_{i,j,r} \to u_{i,j,l}$ (arrived at the left diamond node from the right). The probability of the backward connector is d_j, and the multiedge states back across the diamond has total probability $1 - d_j$.

For the four types of states, we will essentially have two types of transitions: going forwards and going backwards. Note that over all i, the states of the same type and of same depth will have the same transition probabilities. Therefore, when the context is clear, we will often suppress the subscript i in the notation, and analyze the two cases of when i is the target branch versus when i is not the target branch. Finally, there are two special states which never have their logits updated:

1. $u_{i,K,r} \to t_i$: leaf-incoming states for $1 \leq i \leq W$. Since t_i is only adjacent to the node $u_{i,K,r}$, the state $u_{i,K,r} \to t_i$ deterministically transitions to $t_i \to u_{i,K,r}$ (unless t_i is the target, in which case the verifier stops the RL model rollout).

2. f_i: fork-incoming states $u_{i,1,l} \to f$ for $1 \le i \le W$. We fix the transition probabilities of these states to be uniform over the W edge states $f \to u_{i,1,l}$, including the same branch it came from.[1]

We call the next edge states that the transition probabilities a_j, b_j, c_j, d_j above correspond to **desired** states; the other actions are called **undesired** states.

In particular, we have the following:

$$a_j = \frac{\exp\left(\Theta_{R_j^+, L_{j+1}^+}\right)}{\exp\left(\Theta_{R_j^+, L_{j+1}^+}\right) + L\exp\left(\Theta_{R_j^+, L_j^-}\right)}, \qquad b_j = \frac{L\exp\left(\Theta_{R_j^-, L_j^-}\right)}{L\exp\left(\Theta_{R_j^-, L_j^-}\right) + \exp\left(\Theta_{R_j^-, L_{j+1}^+}\right)},$$

$$c_j = \frac{L\exp\left(\Theta_{L_j^+, R_j^+}\right)}{L\exp\left(\Theta_{L_j^+, R_j^+}\right) + \exp\left(\Theta_{L_j^+, R_{j-1}^-}\right)}, \qquad d_j = \frac{\exp\left(\Theta_{L_j^-, R_{j-1}^-}\right)}{\exp\left(\Theta_{L_j^-, R_{j-1}^-}\right) + L\exp\left(\Theta_{L_j^-, R_j^+}\right)}.$$

For a given state s, let a_1 be its desired next state, and a_0 be its undesired next state. Then the pretrained model satisfies $\Theta_{s,a_1} = \Theta_{s,a_0} = 0$ and $\Theta_{s,a'} = -\infty$ for a' not in the set of possible next actions, so that $a_j(0) = d_j(0) = \frac{1}{L+1}$ and $b_j(0) = c_j(0) = \frac{L}{L+1}$.

Finally, we denote $H_f(t) = \mathbb{E}_{y\sim\pi_{\Theta(t)}}[|y|]$ to be the expected hitting time at time t after the start of post-training. Note that for any target $x = t_i$, this value is again the same.

4.2 SFT Learned Model

Theorem 2 (Transitions learned by SFT)**.** *When training with gradient flow on the SFT loss (Equation* (1)*) over only golden paths, it holds that a_j, c_j converge arbitrarily close to* 1 *in finite time. On the other hand, b_j, d_j are fixed at $\frac{L}{L+1}, \frac{1}{L+1}$, respectively, over all time.*

Essentially, this theorem states that all forward-pointing edges will learn to put transition mass towards the next forward edge. On the other hand, because the model is only given golden paths towards the target, all backward pointing edge states remain uniform over its neighbors. In particular, because there is no explicit backtracking data in the fine-tuning dataset, the only ability to backtrack comes from what is learned during the pretraining phase, with no chance to be amplified post-training.

4.3 RLVR Learned Model

Theorem 3 (Transitions learned by RLVR)**.** *When training with sign gradient flow on the RLVR loss (Equation* (2)*), a_j, b_j, c_j, d_j converge arbitrarily close to 1 in finite time.*

Intuitively, on-policy rollouts in RLVR penalizes the model for attempts containing long back-and-forth paths; it needs to amplify its backtracking ability to mitigate this. As such, the learned model converges to the state where all edge states that point forward on a branch will learn to keep going forward, and all edge states that point backwards on a branch will learn to keep going backward.

4.4 Separation in Inference-time Efficiency

Given the policy learned from either SFT or RLVR, our main result shows that the former can be exponentially more inefficient at inference time. This is due to the fact that fine-tuning on only the golden paths induces a lack of backtracking ability for the SFT model, which is well known to memorize the data distribution. For ease of exposition, we suppose that the SFT and RLVR models have converged to the transitions described in Theorem 2 and Theorem 3 (equivalently, the $t = \infty$ limit of the training process).

Theorem 4 (Inference-time separation)**.** *Suppose that the learned RLVR model with $a_j, b_j, c_j, d_j = 1$ and the learned SFT model are prompted with any target node u in the graph to find a path starting from the source. Then the RLVR model requires $\Theta(WK)$ time in expectation to find u, while the SFT model requires $\Theta(WL^K)$ time.*

[1] If these logits are also updated, it is straightforward to see that they will converge to the uniform distribution on the remaining $W - 1$ branches, discarding the branch it has ostensibly already explored. Since this does not affect the final guarantees order-wise, we fix these logits for simplicity.

The above theorem implies a separation of $\Theta(L^K)$, which is exponential in the world model's depth.

4.5 Distilling RLVR Reasoning Traces

Finally, we highlight when SFT can succeed for our reasoning task. Suppose we have access to the reasoning traces of a trained model, such as the outputs of a powerful closed-source model. If we fine-tune a pretrained base model on such traces (a process known as distillation (Shridhar et al., 2022)), we obtain similar guarantees in terms of inference-time efficiency, thereby avoiding the exponential blowup in Theorem 4.

Theorem 5 (SFT on reasoning traces)**.** *Denote the joint distribution of target and reasoning trace pairs generated by the RLVR converged model to be $\mathcal{D}$. Then, if we fine-tune the pretrained model on these traces by optimizing the following loss:*

$$\min_{\Theta} L_{\text{distill}}(\Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[-\sum_{t=0}^{|y|-1}\log \pi_\Theta(y_{t+1}|y_t)\right]$$

we can achieve $\Theta(WK)$ expected inference time compute.

The fact that reasoning traces learned by a reinforcement fine-tuned model are useful for distillation demonstrates the importance of backtracking in fine-tuning data.

5 Experiments

In this section, we run experiments on our synthetic setup to validate our theory. We provide example plots for the case of $W = 15, K = 15, L = 5$, run under sign gradient descent with learning rate 0.01. Figure 3 shows that the final hitting time indeed converges to $\Theta(KW)$, as suggested by Section 4.4. Moreover, Figure 4 shows that all probabilities a_j, b_j, c_j, d_j indeed converge to 1 with a rate (only) depending on their type. While not visually clear, we note that there is a very brief regime in the beginning in which the probabilities b and d are decreasing for middle depth nodes.

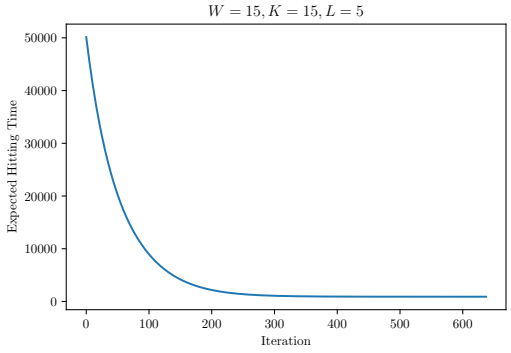

Figure 3: Expected hitting time of RLVR-trained model against training iterations. Here, hitting time converges to $4WK = 900$.

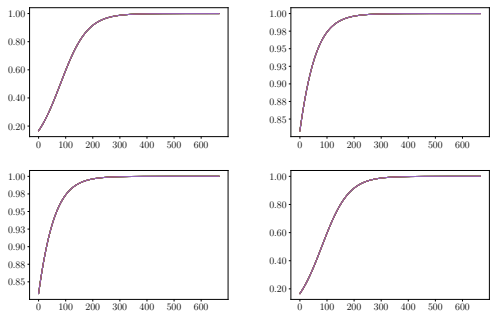

Figure 4: a_j, b_j, c_j, d_j values for RLVR-trained model (ordered in Z shape). Values of edges over all depths j are overlaid. Note that $a(0) = d(0) = \frac{1}{L+1}$ and $b(0) = c(0) = \frac{L}{L+1}$.

6 Conclusion

We highlight backtracking as a key capability for reasoning. We introduce a simple graph pathfinding model of chain-of-thought reasoning and give a dynamical comparison between supervised fine-tuning on shortest-path demonstrations and RL with verifiable rewards. We show that SFT on gold traces need not update backward-state transitions and can incur exponential search cost, whereas RLVR's on-policy rollouts provide signal to learn efficient backtracking from failed paths, yielding a provable inference-time compute separation. Finally, we show that RLVR-generated traces can be distilled via supervised learning to transfer efficient backtracking to a base model.

REFERENCES

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? The locality barrier and inductive scratchpad. In *Advances in Neural Information Processing Systems*, 2024.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, 2025. URL https://arxiv.org/abs/2403.06963.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Hongyi James Cai, Junlin Wang, Xiaoyin Chen, and Bhuwan Dhingra. How much backtracking is enough? exploring the interplay of sft and rl in enhancing llm reasoning, 2025. URL https://arxiv.org/abs/2505.24273.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. SFT or RL? An Early Investigation into Training R1-Like Reasoning Large Vision-Language Models. *arXiv preprint arXiv:2504.11468*, 2025.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. *arXiv preprint arXiv:2501.17161*, 2025a.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025b. URL https://arxiv.org/abs/2501.17161.

Andrew Cohen, Andrey Gromov, Kaiyu Yang, and Yuandong Tian. Spectral journey: How transformers predict the shortest path, 2025. URL https://arxiv.org/abs/2502.08794.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang,

Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025a. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025b. URL https://arxiv.org/abs/2512.02556.

Arvid Frydenlund. The mystery of the pathological path-star task for language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, page 12493–12516. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.695. URL http://dx.doi.org/10.18653/v1/2024.emnlp-main.695.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Mikail Khona, Maya Okawa, Jan Hula, Rahul Ramesh, Kento Nishi, Robert Dick, Ekdeep Singh Lubana, and Hidenori Tanaka. Towards an understanding of stepwise inference in transformers: A synthetic graph navigation model, 2024. URL https://arxiv.org/abs/2402.07757.

Juno Kim, Denny Wu, Jason Lee, and Taiji Suzuki. Metastable Dynamics of Chain-of-Thought Reasoning: Provable Benefits of Search, RL and Distillation. *arXiv preprint arXiv:2502.01694*, 2025.

Team Kimi, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip H. S. Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models, 2025. URL https://arxiv.org/abs/2502.21321.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Jieyi Long. Large language model guided tree-of-thought, 2023. URL https://arxiv.org/abs/2305.08291.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Simon Park, Simran Kaur, and Sanjeev Arora. How does rl post-training induce skill composition? a case study on countdown, 2025. URL https://arxiv.org/abs/2512.01775.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to backtrack: When sequential search limits model reasoning, 2025. URL https://arxiv.org/abs/2504.07052.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. Taming imperfect process verifiers: A sampling perspective on backtracking, 2025. URL https://arxiv.org/abs/2510.03149.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *arXiv preprint arXiv:2405.18512*, 2024.

Shai Shalev-Shwartz and Amnon Shashua. From reasoning to super-intelligence: A search-theoretic perspective, 2025. URL https://arxiv.org/abs/2507.15865.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. *arXiv preprint arXiv:2212.00193*, 2022.

Anikait Singh, Kushal Arora, Sedrick Keh, Jean Mercat, Tatsunori Hashimoto, Chelsea Finn, and Aviral Kumar. Improving the efficiency of test-time search in LLMs with backtracking, 2025. URL https://openreview.net/forum?id=hJ2BCYGvFg.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

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

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms, 2025. URL https://arxiv.org/abs/2506.14245.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025. URL https://arxiv.org/abs/2501.09686.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for boosting reasoning of language models, 2025. URL https://arxiv.org/abs/2502.04404.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning, 2025. URL https://arxiv.org/abs/2506.01347.

A Overview of Proofs

A.1 Proof Sketch of SFT Training

We essentially consider initialization from the pretrained model and analyze the gradient update on the cross-entropy loss. Since only the logits corresponding to a_j, c_j change (as the training data does not contain any states R_j^- or L_j^-), we can reduce the argument into an analysis of an ODE on the logit gaps, and this is sufficient to prove Theorem 2.

A.2 Proof Sketch of RLVR Training

In contrast to the SFT setting, the dynamical analysis for RLVR is significantly more involved as the interaction between rollouts and backtracking comes into play. For each logit, the policy gradient is equivalent to

$$\frac{\partial J}{\partial \Theta_{s,a}} = \mathbb{E}_x[d_x(s)\pi(a|s)A_x(s,a)]$$

where $d_x(s) := \mathbb{E}_y[\sum_{t<\tau} \mathbf{1}\{s_t = s\}]$ is the expected number of times we reach state s before hitting the target x, and $A_x(s,a)$ is the advantage for the policy π for the reward (hitting time) when choosing action a compared to the other actions at state s.

In our analysis of RLVR, this also means that we will treat the previously defined quantity $d_x(\cdot)$ for such states as:

$$d_x(u_{i,j,l} \to u_{i,j,r}) := \sum_{\ell=1}^{L} \mathbb{E}_y\left[\sum_{t<\tau} \mathbf{1}\{s_t = (u_{i,j,l} \overset{(\ell)}{\to} u_{i,j,r})\}\right].$$

The analogous statement holds true for the reverse multiedges as well.

For a fixed target x, define the hitting time $\tau_x := \min\{t \geq 0 : \mathrm{head}(s_t) = x\}$. Also, let $h_x(s) := \mathbb{E}[\tau_x | s_0 = s]$, with absorption $h_x(s) = 0$ if $\mathrm{head}(s) = x$ and the Bellman equation:

$$h_x(s) = 1 + \sum_a \pi(a|s)h_x(a) \quad (\mathrm{head}(s) \neq x).$$

Then the advantage satisfies:

$$A_x(s,a) = \bar{h}_x(s) - h_x(a), \quad \bar{h}_x(s) := \sum_{a'} \pi(a'|s)h_x(a').$$

Also, recall the expected number of visits to a directed-edge state s before hitting x is

$$d_x(s) := \mathbb{E}\left[\sum_{t<\tau_x} \mathbf{1}\{s_t = s\}\right].$$

Again, for multiedge states, we will treat this quantity as an aggregate of all L multiedges. Now, the update for each logit becomes:

$$\begin{aligned}\frac{\partial J}{\partial \Theta_{s,a}} &= \mathbb{E}_x[d_x(s)\pi(a|s)A_x(s,a)] \\ &= \frac{1}{W}\sum_{i=1}^{W} d_{t_i}(s)\pi(a|s)\big(\bar{h}_{t_i}(s) - h_{t_i}(a)\big).\end{aligned}$$

In each type of state, there is a desired action and an undesired action. Our goal is to show that the learned model converges to the state where all edge states that point forward on a branch will learn to keep going forward, and all edge states that point backwards on a branch will learn to keep going backward. The update rule of the logit difference of these two actions are related via the following lemma, which is proved in Section C.1.

Lemma 1. *For a state s, let $\Theta_{s,1}$ correspond to the logit for one of the desired next edge states, and $\Theta_{s,0}$ correspond to the logit for one of the undesird next edge states, and let the logit gap $\mathcal{D}_s := \Theta_{s,1} - \Theta_{s,0}$. Then*

$$\frac{\mathrm{d}\mathcal{D}_s}{\mathrm{d}t} = \mathrm{sgn}(\mathbb{E}_x[d_x(s)(h_x(a_0) - h_x(a_1))]).$$

As a corollary, we can express the updates of the logit differences as follows, where we suppress the subscript i for brevity. The following hold for the four types of states:

1. R_j^+ state: Denote the logit gap as $\mathcal{D}_j^{(a)}$. Then,
$$\frac{\mathrm{d}\mathcal{D}_j^{(a)}}{\mathrm{d}t} = \operatorname{sgn} \underbrace{\mathbb{E}_x\left[d_x(R_j^+) \cdot \left(h_x(L_j^-) - h_x(L_{j+1}^+)\right)\right]}_{=:G_j^{(a)}}.$$

2. R_j^- state: Denote the logit gap as $\mathcal{D}_j^{(b)}$. Then,
$$\frac{\mathrm{d}\mathcal{D}_j^{(b)}}{\mathrm{d}t} = \operatorname{sgn} \underbrace{\mathbb{E}_x\left[d_x(R_j^-) \cdot h_x(L_{j+1}^+) - h_x(L_j^-))\right]}_{=:G_j^{(b)}}.$$

3. L_j^+ state: Denote the logit gap as $\mathcal{D}_j^{(c)}$. Then,
$$\frac{\mathrm{d}\mathcal{D}_j^{(c)}}{\mathrm{d}t} = \operatorname{sgn} \underbrace{\mathbb{E}_x\left[d_x(L_j^+) \cdot \left(h_x(R_{j-1}^-) - h_x(R_j^+)\right)\right]}_{=:G_j^{(c)}}.$$

4. L_j^- state: Denote the logit gap as $\mathcal{D}_j^{(d)}$. Then,
$$\frac{\mathrm{d}\mathcal{D}_j^{(d)}}{\mathrm{d}t} = \operatorname{sgn} \underbrace{\mathbb{E}_x\left[d_x(L_j^-) \cdot \left(h_x(R_j^+) - h_x(R_{j-1}^-)\right)\right]}_{=:G_j^{(d)}}.$$

We divide our analysis into two phases: 1) the time until all of the logit gaps are increasing (i.e., $G_j^{(p)} > 0$ for all $1 \le j \le K$ and $p \in \{a, b, c, d\}$) and 2) convergence from said time towards 1.

RLVR at post-training initialization. Denote the start of RLVR as time $t = 0$. The following lemma characterizes the properties of the gradients at initialization.

Lemma 2. *At $t = 0$, it holds that $G_j^{(a)}, G_j^{(c)} > 0$. However, $G_j^{(b)}, G_j^{(d)}$ may be negative for depths $1 \le l(W, K, L) \le j \le r(W, K, L) \le K - 1$, where $l(\cdot), r(\cdot)$ depend on the specific values of W, K, L.*

The full proof is deferred to the appendix. Intuitively, this can be interpreted as the following. First, forward-oriented states (corresponding to $G_j^{(a)}, G_j^{(c)}$) want to continue forwards, as going backwards would be a waste of movement. Second, backward-oriented states (corresponding to $G_j^{(b)}, G_j^{(d)}$) near the end of a branch want to continue backwards, because they tend to believe that they came from a non-target leaf on the current branch. Third, backward-oriented states near the fork tend to put higher preference on the additional optionality of moving towards the fork as opposed to continuing forward on an uncertain branch. Finally, backward-oriented states in the middle are more "confused," in the sense that it is unclear if they are facing backwards because of backtracking or because of a U-turn from a forward-oriented state; this confusion should eventually be resolved as forward-oriented states converge to putting all mass forwards.

RLVR Phase I. It can first be shown that $G_j^{(a)}, G_j^{(c)}$ are not only positive at initialization, but over all time. This leads to a lower bound on the time that $G_j^{(b)}, G_j^{(d)}$ can be negative.

Lemma 3. *Let $T = \Theta\big(\log \frac{H_0 L}{W^2}\big)$, where $H_0 := H_f(0)$. Then for all $1 \le j \le K$, we have $G_j^{(b)}(T), G_j^{(d)}(T) > 0$.*

The proof of this critical fact relies on a careful decomposition of all $G_j^{(p)}$, with full details deferred to the appendix.

RLVR Phase II. Continuing on, we enter the second phase of the training dynamics after time T. In the setting of Lemma 3, we now consider the additional time needed after time $t = T$ to reach the regime where $\min_j (\min\{a_j, b_j, c_j, d_j\}) \geq 1 - \kappa$ for some $\kappa \ll 1$, for which we prove the following guarantee.

Lemma 4. *Let $T' = \Theta\big(\log \frac{L}{\kappa}\big)$. It holds that*

$$G_j^{(p)}(T + T') \geq 1 - \kappa, \quad \forall p \in \{a, b, c, d\}.$$

This result of this phase follows from the self-reinforcing nature of the dynamics in this regime. Combining the results thus far gives Theorem 3. Full proofs can be found in Section C in the appendix.

A.3 Proof Sketch of Inference Time Separation

Given the construction of the converged model in Theorem 3, the model takes $\Theta(K)$ time to traverse any branch back-and-forth, and traverses W branches on expectation before hitting the target branch. Therefore, the required inference time compute is $\Theta(WK)$ for the RLVR model.

For the converged SFT model in Theorem 2, the backtracking behavior is more complex. Upon entering a branch i, since $a_j = c_j = 1$ for all j, we reach the t_i node in $\Theta(K)$ time. However, exiting the branch is more complicated. Starting from the state R^+_{K+1} with head t_i, let g_i denote the expected time of first entry into R^-_{K+1-i}, and f_i denote the expected time of first entry into L^-_{K+1-i}. Then, we have the following recursion with base case $g_1 = 1$:

$$\begin{aligned} f_i &= \frac{L+1}{L} g_i + \frac{2i-1}{L} + 1, \\ g_i &= (L+1) f_{i-1} + (2i-2)L + 1. \end{aligned}$$

We prove and analyze this recurrence in Section D of the appendix. In particular, the time needed from entry of a wrong branch to getting back to the fork state is $g_{K+1} = O(L^K)$, leading to a total compute of $\Theta(WL^K)$.

B Pretraining and Supervised Fine-Tuning

We start off with the following general lemma for cross-entropy loss.

Lemma 5. *Let $\mathcal{Q}$ be any distribution over pairs (s, a) of current edge-state s and next edge-state a. Define the marginal $d_{\mathcal{Q}}(s) := \mathbb{P}_{(s,a)\sim\mathcal{Q}}[s]$ and the conditional $p_{\mathcal{Q}}(a \mid s) := \mathbb{P}_{(s,a)\sim\mathcal{Q}}[a \mid s]$. Consider the cross-entropy objective*

$$L_{\mathcal{Q}}(\Theta) := \mathbb{E}_{(s,a)\sim\mathcal{Q}}\big[-\log \pi_\Theta(a \mid s)\big].$$

Then for every pair (s, a),

$$\frac{\partial L_{\mathcal{Q}}}{\partial \Theta_{s,a}} = d_{\mathcal{Q}}(s)\Big(\pi_\Theta(a \mid s) - p_{\mathcal{Q}}(a \mid s)\Big).$$

In particular, under gradient flow $\frac{\mathrm{d}\Theta_{s,a}}{\mathrm{d}t} = -\partial L_{\mathcal{Q}}/\partial\Theta_{s,a}$, the dynamics of each row $\Theta_{s,\cdot}$ depends only on data transitions out of s; if $d_{\mathcal{Q}}(s) = 0$ then $\Theta_{s,\cdot}(t)$ remains constant.

We now prove the pretraining convergence result.

Proof of Theorem 1. We first initialize $\Theta(0) = 0$ to be the zero logit matrix. Let $\mathcal{D}$ be the world-model transition distribution described in the main text. For each state $s = u \to v$, let $\mathcal{A}(s) := N_v$ denote the set of valid next edge-states and let $m_s := |\mathcal{A}(s)|$. Under $\mathcal{D}$ we have the conditional target distribution

$$p_{\mathcal{D}}(a \mid s) = \begin{cases} \frac{1}{m_s}, & a \in \mathcal{A}(s), \\ 0, & a \notin \mathcal{A}(s). \end{cases}$$

By Lemma 5, the pretraining gradient flow satisfies, for every s, a,

$$\frac{\mathrm{d}\Theta_{s,a}}{\mathrm{d}t} = -d_{\mathcal{D}}(s)\big(\pi_{\Theta(t)}(a \mid s) - p_{\mathcal{D}}(a \mid s)\big).$$

By assumption, we have that all of the states of the world model have equally likely marginal probability. Hence, for all of the gradient flow logits, we can rescale time by $d_{\mathcal{D}}(s)$, so that:

$$\frac{\mathrm{d}\Theta_{s,a}}{\mathrm{d}t} = -\big(\pi_{\Theta(t)}(a \mid s) - p_{\mathcal{D}}(a \mid s)\big).$$

We now fix any state s. Because $\Theta_{s,a}(0) = 0$ for all a, we have complete symmetry at $t = 0$. Moreover, the target distribution $p_{\mathcal{D}}(\cdot \mid s)$ is invariant under permutations within the two groups $\mathcal{A}(s)$ and $\mathcal{A}(s)^c$. Since the ODE is deterministic and coordinates within each group have identical right-hand sides whenever they are equal, uniqueness of ODE solutions implies that for all $t \geq 0$,

$$\Theta_{s,a}(t) = u_s(t) \;\; \forall a \in \mathcal{A}(s), \qquad \Theta_{s,a}(t) = v_s(t) \;\; \forall a \notin \mathcal{A}(s)$$

for some scalars $u_s(t), v_s(t)$.

Let $M := 2|E|$ be the total number of edge-states, and write $n_s := M - m_s$. Define the total probability mass assigned to valid actions

$$p_s(t) := \sum_{a \in \mathcal{A}(s)} \pi_{\Theta(t)}(a \mid s) = \frac{m_s e^{u_s(t)}}{m_s e^{u_s(t)} + n_s e^{v_s(t)}}.$$

Then each valid action has probability $p_s(t)/m_s$ and each invalid action has probability $(1-p_s(t))/n_s$. Applying Lemma 5 to one representative valid action and one representative invalid action gives

$$\frac{\mathrm{d}u_s}{\mathrm{d}t} = \left(\frac{1}{m_s} - \frac{p_s(t)}{m_s}\right) = \frac{1 - p_s(t)}{m_s}, \qquad \frac{\mathrm{d}v_s}{\mathrm{d}t} = -\frac{1 - p_s(t)}{n_s}.$$

Hence for the logit gap $g_s(t) := u_s(t) - v_s(t)$ we have

$$\frac{\mathrm{d}g_s}{\mathrm{d}t} = (1 - p_s(t))\left(\frac{1}{m_s} + \frac{1}{n_s}\right) \geq 0,$$

so $g_s(t)$ is nondecreasing. Writing $w_s(t) := e^{g_s(t)}$ and using

$$p_s(t) = \frac{m_s w_s(t)}{m_s w_s(t) + n_s} \quad \Rightarrow \quad 1 - p_s(t) = \frac{n_s}{m_s w_s(t) + n_s},$$

we obtain

$$\frac{\mathrm{d}w_s}{\mathrm{d}t} = w_s(t)\frac{\mathrm{d}g_s}{\mathrm{d}t} = \left(\frac{1}{m_s} + \frac{1}{n_s}\right)\frac{n_s w_s(t)}{m_s w_s(t) + n_s} \;\geq\; \frac{1}{m_s},$$

where for the final inequality we have used that $w_s(t) \geq w_s(0) = 1$ and thus $\frac{n_s w_s}{m_s w_s + n_s} \geq \frac{n_s}{m_s + n_s}$ to yield the cancellation. Therefore $w_s(t) \geq 1 + \frac{1}{m_s}t$ and hence

$$1 - p_s(t) = \frac{n_s}{m_s w_s(t) + n_s} \leq \frac{n_s}{m_s\left(1 + \frac{1}{m_s}t\right)} = \frac{n_s}{m_s + t}.$$

Thus the total invalid mass $1 - p_s(t)$ goes to 0 as $t \to \infty$, and the valid mass goes to 1. Because all valid logits remain equal, the distribution over valid actions is uniform at all times:

$$\pi_{\Theta(t)}(a \mid s) = \frac{p_s(t)}{m_s} \quad \forall a \in \mathcal{A}(s).$$

Consequently, for any $\varepsilon > 0$ we may choose $T_s(\varepsilon) := n_s/\varepsilon$ so that for all $t \geq T_s(\varepsilon)$ we have $1 - p_s(t) \leq \varepsilon$, and therefore

$$\max_{a \in \mathcal{A}(s)} \left|\pi_{\Theta(t)}(a \mid s) - \frac{1}{m_s}\right| \leq \frac{\varepsilon}{m_s} \quad \text{and} \quad \sum_{a \notin \mathcal{A}(s)} \pi_{\Theta(t)}(a \mid s) \leq \varepsilon.$$

Since this holds for every state s, pretraining under gradient flow learns the world-model transition kernel up to arbitrary error in finite time, as desired. □

Using a similar technique of analyzing a training with a teacher distribution using cross-entropy loss, we can prove the convergence of the SFT model. In the following section, we provide the proof for Theorem 2.

Proof of Theorem 2. We assume the SFT phase is initialized at the pretrained world model described in the text, i.e., for every state s all valid next-actions have equal logits, and invalid actions have zero probability (equivalently logits $-\infty$). We also assume $\mathcal{D}_x$ is uniform with respect to the targets; this only affects the relative timescales of convergence.

Let $\mathcal{Q}_{\mathrm{SFT}}$ denote the induced distribution over adjacent transition pairs (s, a) obtained by the following procedure: sample a target $x \sim \mathcal{U}(\{t_i\})$, sample a golden path $y \sim \mathcal{D}_x$, and then sample a uniformly random transition (y_{t-1}, y_t) (i.e. each edge connecting diamonds, and each of the L multiedges at a diamond will be sampled uniformly at random).

Then the SFT objective can be written as

$$L_{\mathrm{SFT}}(\Theta) = \mathbb{E}_{(s,a)\sim\mathcal{Q}_{\mathrm{SFT}}}\big[-\log \pi_\Theta(a \mid s)\big].$$

By Lemma 5, for each state s the gradient flow depends only on the conditional $p_{\mathcal{Q}_{\mathrm{SFT}}}(\cdot \mid s)$, and if a state s is never encountered in golden paths then its row never updates.

First, observe that by construction, every golden path is a shortest path from s_0 to some t_i and therefore never contains backtracking. In particular, none of the backward-pointing states $R^-_{i,j}$ or $L^-_{i,j}$ appear in y. Hence for each such backward state s we have $d_{\mathcal{Q}_{\mathrm{SFT}}}(s) = 0$ and Lemma 5 implies $\Theta_{s,\cdot}(t)$ is constant over time. Therefore the associated transition probabilities remain equal to their pretrained values:

$$b_j(t) = b_j(0) = \frac{L}{L+1}, \qquad d_j(t) = d_j(0) = \frac{1}{L+1}.$$

We now analyze the forward-oriented states. First, consider a forward-oriented state $R^+_{i,j}$, each of which have exactly one forward action and L backward actions. On any golden path, whenever we are in state $R^+_{i,j}$ the next state is deterministically the forward connector state $L^+_{i,j+1}$. Therefore the SFT conditional satisfies $p_{\mathcal{Q}_{\mathrm{SFT}}}(\cdot \mid R^+_{i,j})$ is a point mass on the forward connector action.

By symmetry among the L backward multiedges and the symmetric pretrained initialization, their logits remain equal for all time, so the row can be summarized by two scalars: $u(t)$ (logit for the unique forward connector) and $v(t)$ (common logit for each backward multiedge). Then

$$a_j(t) = \frac{e^{u(t)}}{e^{u(t)} + Le^{v(t)}}.$$

Applying Lemma 5 to this row yields (up to time rescaling of $d_{\mathcal{Q}_{\mathrm{SFT}}}(R^+_{i,j})$):

$$\frac{\mathrm{d}u}{\mathrm{d}t} = 1 - a_j(t), \qquad \frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{1 - a_j(t)}{L}.$$

Hence for the gap $g(t) := u(t) - v(t)$ we have

$$\frac{\mathrm{d}g}{\mathrm{d}t} = (1 - a_j(t))\left(1 + \frac{1}{L}\right) \geq 0.$$

Let $w(t) := e^{g(t)}$. Using $a_j(t) = \frac{w(t)}{w(t)+L}$ one computes

$$\frac{\mathrm{d}w}{\mathrm{d}t} = w(t)\frac{\mathrm{d}g}{\mathrm{d}t} = (L+1)\frac{w(t)}{w(t)+L} \geq 1,$$

since $w(t) \geq w(0) = 1$. Therefore $w(t) \geq 1 + t$ and

$$1 - a_j(t) = \frac{L}{w(t)+L} \leq \frac{L}{t+L+1}.$$

Thus for any $\kappa > 0$, taking $t \geq \frac{L}{\kappa} - L - 1$ ensures $a_j(t) \geq 1 - \kappa$.

Finally, we consider a forward-oriented state of type $L^+_{i,j}$. This state has L forward multiedges transitions and one backward transition. On any golden path, whenever we are at $L^+_{i,j}$ the next action is always one of the forward multiedges across $\diamondsuit(u_{i,j,l}, u_{i,j,r})$. Therefore the SFT conditional satisfies $p_{\mathcal{Q}_{\mathrm{SFT}}}(\cdot \mid L^+_{i,j})$ on the backward transition is zero.

Let $\rho_j(t)$ denote the model's probability of taking the backward connector at $L^+_{i,j}$; then $c_j(t) = 1 - \rho_j(t)$. The backward connector logit is pushed down at rate proportional to $\rho_j(t)$, and hence $\rho_j(t) \to 0$ and $c_j(t) \to 1$. Under the additional symmetry assumption that $\mathcal{D}_x$ is invariant under permutations of the L multiedges in each diamond (so each forward multiedge is equally likely in the population loss), the same reduction as above applies: all L forward multiedges share a common logit $u(t)$ and the backward connector has logit $v(t)$, so

$$c_j(t) = \frac{Le^{u(t)}}{Le^{u(t)} + e^{v(t)}} = \frac{Lw(t)}{Lw(t)+1}, \quad w(t) := e^{u(t)-v(t)}.$$

The same calculation gives $\frac{\mathrm{d}w}{\mathrm{d}t} \geq 1/L$, hence $w(t) \geq 1 + t/L$ and

$$1 - c_j(t) = \frac{1}{Lw(t)+1} \leq \frac{1}{t+L+1}.$$

Thus for any $\kappa > 0$, taking $t \geq \frac{1}{\kappa} - L - 1$ ensures $c_j(t) \geq 1 - \kappa$, as desired. □

C RLVR

C.1 Setup and Notations

Recall that:

$$\nabla_\Theta J(\Theta) = \mathbb{E}_x \mathbb{E}_{y\sim\pi}[r(x,y)\nabla_\Theta \log \pi_\Theta(y)] = \mathbb{E}_x \mathbb{E}_{y\sim\pi_\Theta}\left[r(x,y)\sum_{t=0}^{|y|-1} \nabla_\Theta \log \pi_\Theta(s_{t+1}|s_t)\right].$$

In the main text, we claimed that this is equivalent to:

$$\frac{\partial J}{\partial \Theta_{s,a}} = \mathbb{E}_x[d_x(s)\pi(a|s)A_x(s,a)]$$

where $d_x(s) := \mathbb{E}_y[\sum_{t<\tau} \mathbf{1}\{s_t = s\}]$ is the expected number of times we reach state s before hitting the target x, and $A_x(s,a)$ is the advantage for the policy π for the reward (hitting time) when choosing action a compared to the other actions at state s. We prove this claim below.

Lemma 6. *The policy gradient can be expressed as:*

$$\mathbb{E}_x \mathbb{E}_{y\sim\pi}\left[r(x,y)\sum_{t=0}^{|y|-1} \nabla_\Theta \log \pi(s_{t+1}|s_t)\right] = \mathbb{E}_x[d_x(s)\pi(a|s)A_x(s,a)]$$

where

$$A_x(s,a) = \bar{h}_x(s) - h_x(a) \quad \text{where} \quad \bar{h}_x(s) := \sum_{a'} \pi(a'|s)h_x(a').$$

Proof. First, we note that under softmax parameterization, we have

$$\pi_\Theta(a|s) = \frac{\exp(\Theta_{s,a})}{\sum_{a'}\exp(\Theta_{s,a'})}$$

and so

$$\frac{\partial \log \pi_\Theta(a_t|s_s)}{\partial \Theta_{s,a}} = \mathbf{1}\{s_t = s\}(\mathbf{1}\{a_t = a\} - \pi_\Theta(a|s)).$$

In our case, we have that $r(x, y) = 1 - |y|$. We can first simplify by noting that for any fixed t,

$$\mathbb{E}[\nabla_\Theta \log \pi_\Theta(a_t|s_t)|s_t] = \sum_a \pi_\Theta(a|s_t)\nabla_\Theta(a|s_t) = \nabla_\Theta\left(\sum_a \pi_\Theta(a|s_t)\right) = 0.$$

Therefore, we obtain:

$$\begin{aligned}\nabla_\Theta J(\Theta) &= \mathbb{E}_{x,y}\left[\sum_{t=0}^{|y|-1}(1-|y|)\mathbf{1}\{s_t = s\}(\mathbf{1}\{a_t = a\} - \pi_\Theta(a|s))\right] \\ &= -\mathbb{E}_{x,y}\left[\sum_{t=0}^{|y|-1}|y|\mathbf{1}\{s_t = s\}(\mathbf{1}\{a_t = a\} - \pi_\Theta(a|s))\right]\end{aligned}$$

To continue, we note that for any fixed t, it holds that:

$$\mathbb{E}[t\nabla_\Theta \log_\Theta(a|s_t)|s_t] = 0$$

due to a similar argument as before (rewards from the past do not impact the gradient). Therefore, we can rewrite the policy gradient as:

$$\begin{aligned}\nabla_\Theta J(\Theta) &= -\mathbb{E}_{x,y}\left[\sum_{t=0}^{|y|-1}|y|\mathbf{1}\{s_t = s\}(\mathbf{1}\{a_t = a\} - \pi_\Theta(a|s))\right] \\ &= -\mathbb{E}_{x,y}\left[\sum_{t=0}^{|y|-1}(|y|-t)\mathbf{1}\{s_t = s\}(\mathbf{1}\{a_t = a\} - \pi_\Theta(a|s))\right] \\ &= \mathbb{E}_x\big[d_x(s)\pi(a|s)(\bar{h}_x(s) - h_x(a))\big]\end{aligned}$$

as desired. □

We are now ready to prove Lemma 1.

Proof of Lemma 1. First, we note that $\pi(a_0|s) + \pi(a_1|s) = 1$. Then, we observe that:

$$\begin{aligned}\bar{h}_x(s) - h_x(a_1) &= \pi(a_0|s)h_x(a_0) + \pi(a_1|s)h_x(a_1) - h_x(a_1) \\ &= \pi(a_0|s)(h_x(a_0) - h_x(a_1))\end{aligned}$$

Similarly,

$$\bar{h}_x(s) - h_x(a_0) = -\pi(a_1|s)(h_x(a_0) - h_x(a_1))$$

Observe that $\bar{h}_x(s) - h_x(a_1)$ and $\bar{h}_x(s) - h_x(a_0)$ have opposite signs. Combining with Lemma 6, we obtain:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{D}_s}{\mathrm{d}t} &= \operatorname{sgn}\big(\mathbb{E}_x\big[d_x(s)\big[\pi(a_1|s)(\bar{h}_x(s) - h_x(a_1)) - \pi(a_0|s)(\bar{h}_x(s) - h_x(a_0))\big]\big]\big) \\ &= \operatorname{sgn}(2\mathbb{E}_x[d_x(s)\pi(a_1|s)\pi(a_0|s)(h_x(a_0) - h_x(a_1))]) \\ &= \operatorname{sgn}(\mathbb{E}_x[d_x(s)(h_x(a_0) - h_x(a_1))])\end{aligned}$$

as desired. □

To characterize the dynamics of RLVR, we first define the following notations.

Definition 1. *Fix a target x. We define the following quantities:*

1. *Let $H_f := h_x(u_{i,1,l} \to f)$ denote the hitting time of a fixed target x from any of the states whose head is the fork f (including the initial source state $s_0 \to f$); by symmetry they are all equal, hence why we consider a single value.*

2. *For state s not on the branch of x, we define $\tilde{g}(s)$ to be the expected time of hitting the fork, starting at state s. That is, $\tilde{g}(s) := \mathbb{E}[\tau_f]$ where $\tau_f := \min\{t \geq 0 : s_0 = s, \mathrm{head}(s_t) = f\}$. In particular, this means that $h_x(s) = \tilde{g}(s) + H_f$, and we have the absorbing condition $h_x(u_{i,1,l} \to f) = 0$ for branch i not equal to the branch of target x.*

3. *For state s on the branch of x, we define $g(s)$ to be the expected time of hitting either f or x (e.g. both states are absorbing). That is, $g(s) := \mathbb{E}[\tau_{\{f,x\}}]$ where $\tau_{\{f,x\}} := \min\{t \geq 0 : s_0 = s, \mathrm{head} \in \{f, x\}\}$.*

4. *For state s on the branch of x, we define $q(s)$ to be the probability it hits f before x. In particular, this means that $h_x(s) = g(s) + q(s)H_f$.*

Lemma 7. *Define the following q-gaps:*

$$\delta_j := q(L_j^-) - q(L_{j+1}^+), \quad \epsilon_j := q(R_{j-1}^-) - q(R_j^+)$$

Then, it holds that $\delta_j > 0$ and $\epsilon_j > 0$ for all j.

Proof. For Δ_j, this follows from intuitively from the fact that all paths from L_{j+1}^+ to the fork must pass through L_j^-. Hence, the event that a path arrives back at the fork from L_{j+1}^+ is a subset of the event that a path arrives back to the fork from L_j^-. Similar reasoning holds for ϵ_j, and the lemma follows. □

Lemma 8. *Define the following $\tilde{g}$-gaps:*

$$\Delta_j := \tilde{g}(L_{j+1}^+) - \tilde{g}(L_j^-), \quad E_j := \tilde{g}(R_j^+) - \tilde{g}(R_{j-1}^-)$$

Then, it holds that $\Delta_j \geq 2$ and $E_j \geq 2$ for all j.

Proof. For Δ_j, this follows from intuitively from the fact that all paths from L_{j+1}^+ to the fork must pass through L_j^-, which requires at least two moves. A similar reasoning holds for E_j, and the lemma follows. □

We remark that we cannot immediately obtain clean positivity results for the g-gaps (i.e., the expected absorbing time onto $\{f, x\}$ on for states on the branch of a target x). To analyze it, we first define the following quantities.

Definition 2. *Fix a target x, and consider the entry state L_1^+ upon entering any branch from the fork. We define the following for a state s on this same branch:*

1. *When s is on the target branch, define $\mu(s) := \mathbb{E}\left[\sum_{t<\tau_{\{f,x\}}} \mathbf{1}\{s_t = s\} | s_0 = L_1^+\right]$. In other words, μ_s is the expected number of visits to state s during a target branch attempt.*

2. *When s is not on the target branch, define $\tilde{\mu}(s) := \mathbb{E}\left[\sum_{t<\tau_f} \mathbf{1}\{s_t = s\} | s_0 = L_1^+\right]$. In other words, $\tilde{\mu}_s$ is the expected number of visits to state s during a non-target branch attempt until it goes back to f.*

Definition 3. *Fix a target x, and define the success probability of hitting the target upon entering its branch p_{succ} before hitting f. That is,*

$$p_{\mathrm{succ}} := \mathbb{P}\{\mathrm{head}(s_{\tau_{\{f,x\}}}) = x | s_0 = L_1^+\}$$

where L_1^+ is the entry state of the branch of x.

Proposition 1. *Fix a target x. Then, the following hold:*

1. *If s is on the same branch as x, then we have $d_x(s) = \frac{\mu(s)}{p_{\mathrm{succ}}}$.*

2. *If s is on a different branch from x, then we have $d_x(s) = \frac{\tilde{\mu}(s)}{p_{\mathrm{succ}}}$.*

Proof. Note that the probability of success for each branch entry is $\frac{p_{\text{succ}}}{W}$, as one needs to choose the target branch with probability $1/W$ and then succeed with probability p_{succ}. Hence, the total number of fork departures on expectation is W/p_{succ}, so the expected number of entries into any fixed branch is $1/p_{\text{succ}}$. □

Proposition 2. *Define:*

$$\Delta g_j^{(a)} := g(L_j^-) - g(L_{j+1}^+), \quad \Delta g_j^{(c)} := g(R_{j-1}^-) - g(R_j^+)$$

Furthermore, define $\Delta g_j^{(b)} := -\Delta g_j^{(a)}$ *and* $\Delta g_j^{(d)} := -\Delta g_j^{(c)}$*. Then, the following hold:*

1. $G_j^{(a)} = \mathbb{E}_x\big[d_x(R_{i,j}^+) \cdot \big(h_x(L_{i,j}^-) - h_x(L_{i,j+1}^+)\big)\big] = \frac{1}{Wp_{\text{succ}}}\Big[\mu(R_j^+)\Big(\Delta g_j^{(a)} + \delta_j H_f\Big) - (W-1)\tilde{\mu}(R_j^+)\Delta_j\Big]$
2. $G_j^{(b)} = \mathbb{E}_x\big[d_x(R_{i,j}^-) \cdot \big(h_x(L_{i,j+1}^+) - h_x(L_{i,j}^-)\big)\big] = \frac{1}{Wp_{\text{succ}}}\Big[\mu(R_j^-)\Big(-\Delta g_j^{(a)} - \delta_j H_f\Big) + (W-1)\tilde{\mu}(R_j^-)\Delta_j\Big]$
3. $G_j^{(c)} = \mathbb{E}_x\big[d_x(L_{i,j}^+) \cdot \big(h_x(R_{i,j-1}^-) - h_x(R_{i,j}^+)\big)\big] = \frac{1}{Wp_{\text{succ}}}\Big[\mu(L_j^+)\Big(\Delta g_j^{(c)} + \epsilon_j H_f\Big) - (W-1)\tilde{\mu}(L_j^+)E_j\Big]$
4. $G_j^{(d)} = \mathbb{E}_x\big[d_x(L_{i,j}^-) \cdot \big(h_x(R_{i,j}^+) - h_x(R_{i,j-1}^-)\big)\big] = \frac{1}{Wp_{\text{succ}}}\Big[\mu(L_j^-)\Big(-\Delta g_j^{(c)} - \epsilon_j H_f\Big) + (W-1)\tilde{\mu}(L_j^-)E_j\Big]$

Proof. We will first consider the case $G_j^{(a)}$. If R_j^+ is on the target branch (which happens with probability $1/W$), then we have:

$$\begin{aligned} d_x(R_j^+) \cdot \big(h_x(L_{i,j}^-) - h_x(L_{i,j+1}^+)\big) &= \frac{\mu(R_j^+)}{p_{\text{succ}}} \cdot \big(\big(g(L_j^-) - g(L_{j+1}^+)\big) + (q(L_j^-) - q(L_{j+1}^+))H_f\big) \\ &= \frac{\mu(R_j^+)}{p_{\text{succ}}} \cdot \Big(\Delta g_j^{(a)} + \delta_j H_f\Big) \end{aligned}$$

If R_j^+ is not on the target branch (which happens with probability $(W-1)/W$), then we have:

$$d_x(R_j^+) \cdot \big(h_x(L_{i,j}^-) - h_x(L_{i,j+1}^+)\big) = \frac{\tilde{\mu}(R_j^+)}{p_{\text{succ}}} \cdot \big(\tilde{g}(L_j^-) - \tilde{g}(L_{j+1}^+)\big) = \frac{\tilde{\mu}(R_j^+)}{p_{\text{succ}}} \cdot (-\Delta_j)$$

This gives the desired expectation. Analogous calculations can be done for the other three cases to obtain the same result. □

We now observe that we can rescale time in the four logit ODE's by a factor of $\frac{2}{Wp_{\text{succ}}}$, since those terms do not change the sign in our gradient flow. Therefore, we redefine:

1. $G_j^{(a)} := \Big[\mu(R_j^+)\Big(\Delta g_j^{(a)} + \delta_j H_f\Big) - (W-1)\tilde{\mu}(R_j^+)\Delta_j\Big]$
2. $G_j^{(b)} := \Big[\mu(R_j^-)\Big(-\Delta g_j^{(a)} - \delta_j H_f\Big) + (W-1)\tilde{\mu}(R_j^-)\Delta_j\Big]$
3. $G_j^{(c)} := \Big[\mu(L_j^+)\Big(\Delta g_j^{(c)} + \epsilon_j H_f\Big) - (W-1)\tilde{\mu}(L_j^+)E_j\Big]$
4. $G_j^{(d)} := \Big[\mu(L_j^-)\Big(-\Delta g_j^{(c)} - \epsilon_j H_f\Big) + (W-1)\tilde{\mu}(L_j^-)E_j\Big]$

so that for $p_j \in \{a_j, b_j, c_j, d_j\}$, the gradient flow is

$$\frac{\mathrm{d}\mathcal{D}_j^{(p)}}{\mathrm{d}t} = \operatorname{sgn}\Big(G_j^{(p)}\Big).$$

C.2 Summary of Important Quantities

For ease of exposition, we summarize the notations defined above in a concise list with its correspondence to branch type.

- Off-target branch: $\tilde{\mu}(\cdot), \tilde{g}(\cdot), \Delta_j, E_j$
- On-target branch: $\mu(\cdot), g(\cdot), \Delta g_j^a, \Delta g_j^c, q(\cdot), \delta_j, \epsilon_j, p_{\text{succ}}$
- Overall: H_f, which is the expected hitting time of fixed target t_i from a branch. Since the targets t_i are symmetric, we have that $J(\Theta) = 1 - H_f$. Moreover, using Proposition 1 yields $H_f = \frac{W+g(L_1^+)+(W-1)\tilde{g}(L_1^+)}{p_{\text{succ}}}$.

Note that in the context of the population loss, the expectation is uniform over the choice of on target branch. In the remainder of this section, we will calculate these quantities exactly in terms of arbitrary a_j, b_j, c_j, d_j.

We give closed form expressions for all of the below quantities, which were calculated with the assistance of SymPy. We verify separately that these quantities (which are by definition unique) indeed hold for the stochastic system we have defined thus far. Unless denoted otherwise, the quantities below hold for all $1 \le j \le K$, with the convention that a state R_0^- is a fork-head state, and L_{K+1}^+ is a leaf-head state, as well as the summation and product of an empty set being 0 and 1 respectively.

C.2.1 Off-target quantities

Define the quantity $r_j := \frac{a_j c_j}{b_j d_j}$.

1. $\tilde{\mu}(\cdot)$: We have
$$\tilde{\mu}(L_j^+) = \prod_{m=1}^{j-1} r_m,$$
$$\tilde{\mu}(R_j^+) = \tilde{\mu}(L_j^-) = \frac{c_j}{d_j} \prod_{m=1}^{j-1} r_m,$$
$$\tilde{\mu}(R_j^-) = \tilde{\mu}(L_{j+1}^+) = \prod_{m=1}^{j} r_m.$$
2. E_j: We have
$$E_j = \sum_{m=j}^{K} \left(\frac{2(a_m + b_m)}{b_m d_m} \prod_{i=j}^{m-1} \frac{a_i c_{i+1}}{b_i d_i} \right).$$
3. Δ_j: We have
$$\Delta_j = \frac{2 + c_{j+1} E_{j+1}}{b_j} = \frac{d_j E_j - 2}{a_j}.$$
4. $\tilde{g}(\cdot)$: Define $F_j := \tilde{g}(R_j^-)$, so that $F_0 = 0$. Then we have
$$F_j = \sum_{m=1}^{j} (2 + (1 - d_m) E_m + (1 - b_m) \Delta_m)$$
and
$$\tilde{g}(R_j^-) = F_j,$$
$$\tilde{g}(R_j^+) = F_{j-1} + E_j,$$
$$\tilde{g}(L_j^+) = 1 + F_{j-1} + c_j E_j,$$
$$\tilde{g}(L_j^-) = 1 + F_{j-1} + (1 - d_j) E_j.$$

We will generally not be working with $\tilde{g}$ directly, but rather their adjacent differences E_j, Δ_j.

C.2.2 ON-TARGET QUANTITIES

Define the following quantities:

$$
\begin{aligned}
r_j &:= \frac{a_j c_j}{b_{j-1} d_j}, \quad \forall\, 2 \le j \le K, \\
P_j &:= \prod_{m=j+1}^{K} r_m, \\
\alpha_j &:= (1 - a_j) + \frac{a_j}{d_j}(1 - c_j), \\
S_j &:= \sum_{m=j}^{K} \alpha_m P_m, \\
Z_q &:= P_1 \left(1 - a_1 + \frac{a_1}{d_1}\right) + S_2.
\end{aligned}
$$

We also have the following quantities.

1. δ_j: We have $\delta_j = P_j / Z_q$.
2. ϵ_j: We have $\epsilon_j = \delta_j a_j / d_j$.
3. Δg_j^a: First, define the following:

$$
\begin{aligned}
\mathcal{B}_j &:= -\sum_{m=j+1}^{K} \frac{2(c_m + d_m)}{b_{m-1} d_m} \prod_{i=j+1}^{m-1} r_i, \\
\beta_m &:= (1 - a_m) + \frac{a_m (1 - c_m)}{d_m}, \\
\gamma_m &:= 2 - \frac{2(1 - c_m)}{d_m}, \\
P_g &:= \sum_{m=2}^{K} \beta_m P_m, \\
Q_g &:= \sum_{m=2}^{K} (\beta_m \mathcal{B}_m + \gamma_m).
\end{aligned}
$$

   Then, it holds that:

$$
\begin{aligned}
\Delta g_K^a &= -\frac{\left(\frac{a_1}{d_1} + 1 - a_1\right)\mathcal{B}_1 - \frac{2}{d_1} + Q_g + 1}{\left(\frac{a_1}{d_1} + 1 - a_1\right) P_1 + P_g}, \\
\Delta g_j^a &= P_j \Delta g_K^a + \mathcal{B}_j.
\end{aligned}
$$

4. Δg_j^c: We have

$$
\Delta g_j^c = \frac{a_j \Delta g_j^a - 2}{d_j}.
$$

   Equivalently, when $2 \le j \le K$,

$$
\Delta g_j^c = \frac{b_{j-1} \Delta g_{j-1}^a + 2}{c_j}.
$$

5. $g(\cdot)$: For simplicity, we will not write down the full expression, since we will generally be working with adjacent differences Δg_j^a and Δg_j^c.

6. $\mu(\cdot)$: We define the prefix product $A_j := \prod_{m=1}^{j} a_m$, as well as B_j, C_j, D_j similarly. Then,

$$\mu(L_j^+) = \frac{A_{j-1}C_{j-1}}{B_{j-1}D_{j-1}} \cdot \frac{S_j + \frac{a_j c_j}{d_j} P_j}{Z_q},$$
$$\mu(R_j^+) = \frac{A_{j-1}C_j}{B_{j-1}D_j} \cdot \frac{S_{j+1} + P_j}{Z_q},$$
$$\mu(L_j^-) = \mu(R_j^+) - p_{\text{succ}},$$
$$\mu(R_j^-) = \mu(L_{j+1}^+) - p_{\text{succ}}.$$

C.2.3 Closed Forms for $G_j^{(p)}$

The aforementioned quantities are also sufficient for us to write a closed form for $G_j^{(p)}$ for $p \in \{a, b, c, d\}$. As before, define $A_j = \prod_{m=1}^{j} a_j$ to be the prefix product of a, and similarly for B_j, C_j, D_j. In particular, we have:

$$G_j^{(a)} = \frac{A_{j-1}C_j}{B_{j-1}D_j} \cdot \frac{S_{j+1} + P_j}{Z_q} \left[P_j \Delta g_K^a + \mathcal{B}_j + \frac{P_j}{Z_q} H_f \right] - (W-1) \frac{A_{j-1}C_j}{B_{j-1}D_j} \Delta_j,$$
$$G_j^{(b)} = \left[\frac{A_j C_j}{B_j D_j} \cdot \frac{S_{j+1} + b_j P_j}{Z_q} - p_{\text{succ}} \right] \cdot \left[-P_j \Delta g_K^a - \mathcal{B}_j - \frac{P_j}{Z_q} H_f \right] + (W-1) \frac{A_j C_j}{B_j D_j} \Delta_j,$$
$$G_j^{(c)} = \left[\frac{A_{j-1}C_{j-1}}{B_{j-1}D_{j-1}} \cdot \frac{S_j + b_{j-1} P_{j-1}}{Z_q} \right] \cdot \left[\frac{a_j (P_j \Delta g_K^a + \mathcal{B}_j) - 2}{d_j} + \frac{a_j}{d_j} \frac{P_j}{Z_q} H_f \right] - (W-1) \frac{A_{j-1}C_{j-1}}{B_{j-1}D_{j-1}} E_j,$$
$$G_j^{(d)} = \left[\frac{A_{j-1}C_j}{B_{j-1}D_j} \cdot \frac{S_{j+1} + P_j}{Z_q} - p_{\text{succ}} \right] \cdot \left[-\frac{a_j (P_j \Delta g_K^a + \mathcal{B}_j) - 2}{d_j} - \frac{a_j}{d_j} \frac{P_j}{Z_q} H_f \right] + (W-1) \frac{A_{j-1}C_j}{B_{j-1}D_j} E_j.$$

C.2.4 Values at Post-training Initialization

We calculate these values at initialization of post-training, in which $a_j = d_j = \frac{1}{L+1}$ and $b_j = c_j = \frac{L}{L+1}$ for all j. Let us denote $D := 1 + K + \frac{K}{L}$. We will not explicitly show the calculations below, with the understanding that they can be obtained from the closed form expressions from the previous section evaluated at these a_j, b_j, c_j, d_j.

First note that at initialization ($t = 0$), it holds that $p_{\text{succ}} = 1/D$. Now consider the values below at initialization. On the target branch i, we have for $1 \le j \le K$:

1. $d_x(L_{i,j}^+) = (K - j + 2) + \frac{K-j+1}{L}$,
2. $d_x(R_{i,j}^-) = (K - j) + \frac{K-j}{L}$,
3. $d_x(R_{i,j}^+) = \left[(K - j + 1) + \frac{K-j+1}{L} \right] \cdot L$,
4. $d_x(L_{i,j}^-) = \left[(K - j + 1) + \frac{K-j}{L} \right] \cdot L = \left[d_x(R_{i,j}^+) - \frac{1}{L} \right] \cdot L$.

Moreover,

1. On a non-target branch, it holds that $d_x(L_j^+) = d_x(R_j^-) = D$ and $d_x(R_j^+) = d_x(L_j^-) = D \cdot L$.
2. The hitting time from a fork state is $H_0 := H_f(0) = (2W - 1)D(1 + K(L + 1))$.

For all states s, it holds at initialization ($t = 0$) that $\tilde{\mu}(s) = 1$ for non-multiedge states, and $\mu(s) = L$ otherwise. That is, $\mu(L_j^-) = \mu(R_j^+) = L$ and $\mu(L_j^+) = \mu(R_j^-)$.

For all $1 \le j \le K$, it holds that at initialization ($t = 0$):

$$\tilde{g}(L_j^+) = \tilde{g}(L_j^-) = j + (L+1)j(2K + 1 - j) + \frac{j-1}{L}((L+1)(2K + 1 - j) + 1),$$
$$\tilde{g}(R_j^+) = \tilde{g}(R_j^-) = j + (L+1)j(2K + 1 - j) + \frac{j}{L}((L+1)(2K - j) + 1).$$

Hence, we obtain:

$$\Delta_j(0) = \tilde{g}(L_{j+1}^+) - \tilde{g}(L_j^-) = \frac{2(L+1)}{L}((L+1)(K-j)+1),$$
$$E_j(0) = \tilde{g}(R_j^+) - \tilde{g}(R_{j-1}^-) = \frac{2(L+1)}{L}((L+1)(K+1-j)).$$

For all $1 \le j \le K$, it holds that at initialization ($t = 0$):

$$\mu(R_j^+) = \left[\frac{1}{D}\left((K+1-j) + \frac{K+1-j}{L}\right)\right] \cdot L,$$
$$\mu(L_j^-) = \left[\frac{1}{D}\left((K+1-j) + \frac{K-j}{L}\right)\right] \cdot L,$$
$$\mu(L_j^+) = \frac{1}{D}\left((K+2-j) + \frac{K+1-j}{L}\right),$$
$$\mu(R_j^-) = \frac{1}{D}\left((K-j) + \frac{K-j}{L}\right).$$

For all $1 \le j \le K$, it holds that at initialization ($t = 0$):

$$g(L_j^+) = g(L_j^-) = (L+1)(K+1-j)\left(j + \frac{j-1}{L}\right),$$
$$g(R_j^+) = g(R_j^-) = (L+1)\left(j(K+1-j) + \frac{j(K-j)}{L}\right).$$

Hence, we have that:

$$\Delta g_j^{(a)} = g(L_j^-) - g(L_{j+1}^+) = -(L+1)\left((K-2j) + \frac{K-2j+1}{L}\right),$$
$$\Delta g_j^{(c)} = g(R_{j-1}^-) - g(R_j^+) = -(L+1)\left((K+2-2j) + \frac{K-2j+1}{L}\right).$$

For all $1 \le j \le K$, it holds that:

$$q(L_j^+)(0) = q(L_j^-)(0) = \frac{K-j+1+\frac{K-j+1}{L}}{D}, \quad q(R_j^-)(0) = q(R_j^-)(0) = \frac{K-j+1+\frac{K-j}{L}}{D}.$$

Hence, we obtain:

$$\delta_j(0) = q(L_j^-)(0) - q(L_{j+1}^+)(0) = \frac{1}{D} \cdot \frac{L+1}{L},$$
$$\epsilon_j(0) = q(R_{j-1}^-) - q(R_j^+) = \frac{1}{D} \cdot \frac{L+1}{L}.$$

Given these quantities, we can now evaluate $G_j^{(p)}$ for $p \in \{a, b, c, d\}$, allowing us to show Lemma 2.

Proof of Lemma 2. We first gives the values of the $G_j^{(p)}$ at initialization. For all depths $1 \le j \le K$, it holds at initialization ($t = 0$) that:

$$G_j^{(a)}(0) = \frac{2(L+1)}{L^2W}\big((L+1)^2 j(K-j) + j\big(W(L^2-1) + 2(L+1)\big) + (W-1)((L+1)K+1)\big),$$
$$G_j^{(b)}(0) = \frac{2(L+1)}{L^2W}\big((L+1)^2 j^2 - (L+1)(K(L+1) + (L-1)(W-1))j + L(W-1)(KL+K+1)\big),$$
$$G_j^{(c)}(0) = \frac{2(L+1)}{L^2W}\big(W(KL+K+L) + (L+1)^2(j-1)(K-j) + (L+1)(j-1)(LW+L-W+1)\big),$$
$$G_j^{(c)}(0) = \frac{2(L+1)}{L^2W}\big((L+1)^2 j^2 - (L+1)((K+1)(L+1) + W(L-1))j + LW(KL+K+L)\big).$$

It is easy now to check that $G_j^{(a)}$ and $G_j^{(c)}$ are positive at initialization, whereas the j in the middle depths can possibly cause $G_j^{(b)}$ and $G_j^{(d)}$ to be negative. □

C.3 Dynamics of Phase I of RLVR

For Lemma 3, we claimed that when $G_j^{(a)}$ and $G_j^{(c)}$ are positive over all time, there is a finite time before $G_j^{(b)}$ and $G_j^{(d)}$ all become positive. We prove Lemma 3 below.

Proof of Lemma 3. Suppose that $G_j^{(a)}$ and $G_j^{(c)}$ are always positive; that is, the probabilities a and c are always increasing. Then, we can express in closed form the following:

$$a(t) = \frac{e^{2t}}{e^{2t} + L}, \qquad c(t) = \frac{Le^{2t}}{Le^{2t} + 1}.$$

Let $T := \frac{1}{2} \log \frac{2H_0^3 L}{W(W-1)}$. We claim that at some time $t \leq T$, all of $G_j^{(b)}$ and $G_j^{(d)}$ become positive, and continue to remain positive. We will prove this is the case first for $G_j^{(b)}$.

Recall that:

$$G_j^{(b)} > 0 \iff (W-1)\tilde{\mu}(R_j^-)\Delta_j > \mu(R_j^-)(\Delta g_j^a + \delta_j H_f).$$

First, observe that $\tilde{\mu}(R_j^-) \geq \mathbb{P}\{R_j^- \text{ is reached}\} \geq p_{\text{succ}}$, because reaching the leaf of a non-target branch implies that R_j^- must have been reached on the way back to the fork. In addition, we have:

$$H_0 \geq H_f = \frac{W + g(L_1^+) + (W-1)\tilde{g}(L_1^+)}{p_{\text{succ}}} \implies p_{\text{succ}} \geq \frac{W}{H_0}.$$

Combining this with Lemma 8, we obtain

$$(W-1)\tilde{\mu}(R_j^-)\Delta_j \geq \frac{2W(W-1)}{H_0}.$$

Moreover, we have that $\Delta g_j^a + \delta_j H_f \leq 2H_f \leq 2H_0$, since it is the difference of leaf hitting times on a target branch, which is trivially bounded by their sum. We can also upper bound $\mu(R_j^-)$ by the expected number of adjacent states in the target branch process that are forwards followed by backwards. In particular, the probability of this for any given state is upper bounded by $\max_j\{\max(1-a_j, 1-c_j)\}$, and we know that the expected attempt length on the target state is bounded by $H_f \leq H_0$. Under our assumption that a_j, c_j are always increasing, this implies that:

$$\mu(R_j^-) \leq H_0 \max_j\{\max(1-a_j, 1-c_j)\} \leq H_0 \cdot \frac{L}{e^{2t} + L} \leq H_0 L e^{-2t}.$$

Combining these bounds, we obtain

$$\mu(R_j^-)(\Delta g_j^a + \delta_j H_f) \leq H_0 L e^{-2t} \cdot 2H_0 = 2H_0^2 L e^{-2t}.$$

For our choice of time T, it holds that

$$(W-1)\tilde{\mu}(R_j^-)\Delta_j \geq \frac{2W(W-1)}{H_0} \geq 2H_0^2 L e^{-2t} \geq \mu(R_j^-)(\Delta g_j^a + \delta_j H_f)$$

which proves that for $t \geq T$, we have $G_j^{(b)}(t) > 0$.

A similar argument for $G_j^{(d)}$ applied on the corresponding terms yields the same time bound, and this concludes the proof. □

The above lemma relies on the fact that $G_j^{(a)}$ and $G_j^{(c)}$ are positive for all time. Under certain mild assumptions on W, K, L, we can show this remains true until the time in which b_j, d_j become increasing.

Assumption 2. *Assume that $K \gg \log \frac{H_0}{L}$, and suppose that the left threshold of positive $G_j^{(b)}$ at initialization satisfies $l(W, K, L) \gg \log \frac{H_0}{L}$.*

Lemma 9. *For $j \geq l(W, K, L)$, it holds that by time $t_j = \frac{1}{4j} \log \frac{H_0}{L}$, $G_j^{(b)}$ and $G_j^{(d)}$ become positive. Moreover, $G_j^{(a)}, G_j^{(c)}$ do not drop below zero in this timeframe.*

Proof. First, in this regime, $a_j(t) = \frac{e^{2t}}{e^{2t}+L}$ and $c_j(t) = \frac{Le^{2t}}{Le^{2t}+1}$ for any j; we note that in this time, the logits have not moved much and hence it holds that $a_j/d_j = \Theta(1)$ and similarly $b_j/c_j = \Theta(1)$. Moreover, it can be shown that $G_j^{(a)}$ and $G_j^{(c)}$ cannot move far from initialization, so it cannot become negative; this can be verified via the calculations in the previous sections.

Let us now fix j. First, recall that $G_j^{(b)} > 0$ if and only if:

$$(W-1)\tilde{\mu}(R_j^-)\Delta_j > \mu(R_j^-)(\Delta g_j^a + \delta_j H_f)$$

Note that the left hand side is positive, so it suffices to show that within time $\frac{1}{4j} \log \frac{H_0}{L}$, the right hand side becomes negative. To begin, let us upper bound $\mu(R_j^-)$.

$$\begin{aligned}
\mu(R_j^-) &= \mu(L_{j+1}^+) - p_{\text{succ}} \\
&\leq \mu(L_{j+1}^+) \\
&= \frac{A_j C_j}{B_j D_j} \cdot \frac{S_{j+1} + \Theta(1) P_{j+1}}{Z_q} \\
&\leq e^{4tj} \cdot \frac{S_{j+1} + \Theta(1) P_{j+1}}{Z_q} \\
&\lesssim e^{4tj} \cdot \frac{e^{4t(K-j)}}{Z_q} \\
&\lesssim e^{4tj} \cdot \frac{e^{4t(K-j)}}{K}
\end{aligned}$$

where we used the fact that $d_j(t) \geq \frac{e^{-2t}}{e^{-2t}+L}$ and $b_j(t) \geq \frac{Le^{-2t}}{Le^{-2t}+1}$, which means $\frac{a_j c_j}{b_j d_j} \leq e^{4t}$ after manipulation. We also used fact that $Z_q \geq P_1 + S_2 \geq K$.

We now proceed to bound the other term on the right hand side:

$$\begin{aligned}
\Delta g_j^a + \delta_j H_f &= P_j \Delta g_K^a + \mathcal{B}_j + \delta_j H_f \\
&\leq e^{4t(K-j)} \Delta g_K^a + \mathcal{B}_j + \delta_j H_f \\
&\leq e^{4t(K-j)} \frac{\frac{2}{d_1}(-Le^{4tK}) + Le^{4t(K-j)}}{K} + \mathcal{B}_j + \delta_j H_f \\
&\leq e^{4t(K-j)} \frac{\frac{2}{d_1}(-Le^{4tK}) + Le^{4t(K-j)}}{K} + Le^{4t(K-j)} + \delta_j H_f
\end{aligned}$$

where we used the fact that $S_j \lesssim e^{4t(K-j)}$, and

$$\begin{aligned}
\Delta g_j^a &\leq \frac{-\left(\frac{a_1}{d_1} + 1 - a_1\right)\mathcal{B}_1 + \frac{2}{d_1} - Q_g - 1}{(\frac{a_1}{d_1} + 1 - a_1)P_1 + P_g} \\
&\leq \frac{\frac{2}{d_1} - Le^{4tK} + Le^{4t(K-j)}}{K}
\end{aligned}$$

due to the fact that $\mathcal{B}_j \asymp -Le^{4t(K-j)}$.

From here, we observe that a sufficient condition for the right hand side to be negative is:

$$\begin{aligned}
&\mu(R_j^-)(\Delta g_j^a + \delta_j H_f) < 0 \\
\Longleftarrow\ &-\frac{L}{K^2}e^{8tK} + \frac{L}{K^2}e^{4tK} + \frac{L}{K^2}e^{8tK-4tj} + \left(\frac{L}{K} + \frac{H_f}{K^2}\right)e^{8tK-4tj} < 0 \\
\Longleftarrow\ &\frac{H_f}{K^2}e^{8tK-4tj} < \frac{L}{K^2}e^{8tK} \\
\Longleftarrow\ &t > \frac{1}{4j}\log\frac{H_0}{L}
\end{aligned}$$

where in the last line we used the fact that $H_f(t) \le H_0$, and the lemma follows for $G_j^{(b)}$. A similar analysis on the corresponding terms of $G_j^{(d)}$ gives an analogous result for that case too. □

C.4 Dynamics of Phase II of RLVR

In this section, we show that once all the logits for a, b, c, d over all depths are increasing, they will continue to increase towards 1.

Proof of Lemma 4. Fix a desired accuracy $\kappa \ll 1$. Note that $a \ge d$ and $c \ge b$ at all time, so it suffices to consider the backwards state b_j, d_j. This is because we work with signed gradient descent, and hence the logits for b_j and d_j will always be smaller than c_j and a_j, respectively.

For $b_j \ge \kappa$ to hold, we require:

$$\begin{aligned}
b_j = \frac{Le^{\mathcal{D}_j^{(b)}}}{Le^{\mathcal{D}_j^{(b)}} + 1} &\ge 1 - \kappa \\
\iff Le^{\mathcal{D}_j^{(b)}} &\ge (1-\kappa)(Le^{\mathcal{D}_j^{(b)}} + 1) \\
\iff \mathcal{D}_j^{(b)} &\ge \log\frac{1-\kappa}{\kappa L}
\end{aligned}$$

Similarly for $d_j \ge \kappa$ to hold, we require:

$$\begin{aligned}
d_j = \frac{e^{\mathcal{D}_j^{(d)}}}{e^{\mathcal{D}_j^{(d)}} + L} &\ge 1 - \kappa \\
\iff e^{\mathcal{D}_j^{(d)}} &\ge (1-\kappa)(e^{\mathcal{D}_j^{(d)}} + L) \\
\iff \mathcal{D}_j^{(d)} &\ge \log\frac{L(1-\kappa)}{\kappa}
\end{aligned}$$

Since $\log\frac{L(1-\kappa)}{\kappa} \ge \log\frac{1-\kappa}{L\kappa}$, it suffices to run for time $\log\frac{L(1-\kappa)}{\kappa}$ after $G_j^{(b)}$ and $G_j^{(d)}$ become positive, as desired. □

Proof of Theorem 3. The theorem follows directly from the conclusion of Lemma 4. □

D Inference Time Separation

In this section, we will prove the separation in inference time compute needed for the SFT model vs. the RLVR model, as formalized by Theorem 4.

Proof of Theorem 4. As noted in Section A.3, the RLVR model requires only $\Theta(WK)$ time to find a path. For the converged SFT model in Theorem 2, we recall the following properties of the backtracking behavior that were described in the main text.

1. Upon entering a branch i, since $a_j = c_j = 1$ for all j, we reach the t_i node in $\Theta(K)$ time.

2. When exiting the branch, starting from the state R^+_{K+1} (i.e. reached the state with head t_i), let g_i denote the expected time of first entry into R^-_{K+1-i}, and f_i denote the expected time of first entry into L^-_{K+1-i}. Then, with the base case of $g_1 = 1$, we have the following recursion:

$$f_i = \frac{L+1}{L} g_i + (2i-1) \cdot \frac{1}{L} + 1$$
$$g_i = (L+1) f_{i-1} + (2i-2) \cdot L + 1$$

Hence, the time needed from entry of a wrong branch to getting back to the fork state is g_{K+1}.

We now justify our recurrence equations. Suppose we are currently at the state R^-_{K+1-i}, and we are trying to reach the next state L^-_{K+1-i}. There is a $L/(L+1)$ chance that we proceed forward, and a $1/(L+1)$ we make a u-turn and end up at the leaf again (which takes $2i-1$ time). By standard geometric mean properties, we must make on expectation $(L+1)/L$ attempts before we can successfully get from R^-_{K+1-i} to L^-_{K+1-i}. This tells us that on the first $\frac{L+1}{L} - 1 = \frac{1}{L}$ attempts, we must pay an extra cost of $2i-1$ to go back to the leaf. A similar analysis holds for the case of L^-_{K+1-i}, where the immediate goal is to reach R^-_{K-i}. Here, we must make on expectation $L+1$ attempts, which means the first $L+1-1 = L$ attempts pay an extra $2i-2$ to get back to the leaf.

We will proceed to write down a closed form for g_{K+1}. First, note that:

$$f_{i-1} = \frac{L+1}{L} g_{i-1} + (2i-1) \cdot \frac{1}{L} + 1$$

Substituting this into the expression for g_i, we have:

$$\begin{aligned} g_i &= (L+1) f_{i-1} + (2i-2) \cdot L + 1 \\ &= (L+1)\left(\frac{L+1}{L} g_{i-1} + (2i-1) \cdot \frac{1}{L} + 1 \right) + (2i-2) \cdot L + 1 \\ &= \frac{(L+1)^2}{L} g_{i-1} + i \cdot \frac{2(L^2+L+1)}{L} - \frac{L^2+L+3}{L} \end{aligned}$$

Let $r = \frac{(L+1)^2}{L}$, $\alpha = \frac{2(L^2+L+1)}{L}$, and $\beta = -\frac{L^2+L+3}{L}$. Then, our recurrence becomes $g_i = r g_{i-1} + \alpha i + \beta$. By an induction argument, we obtain that:

$$g_{K+1} = r^K + \alpha \sum_{t=2}^{K+1} t r^{K+1-t} + \beta \sum_{t=2}^{K+1} r^{K+1-t}$$

The first term is of course at least L^K. For the second term, the $t = 2$ term multiplied by α already contributes a term of at least $2L^K$. For the third term, it is simply a geometric series, which can be evaluated to be:

$$\beta \cdot \frac{r^K - 1}{r - 1} \geq -2(r^K - 1)$$

for $L \geq 2$. Combining everything, we obtain that

$$g_{K+1} = r^K + \alpha \sum_{t=2}^{K+1} t r^{K+1-t} + \beta \sum_{t=2}^{K+1} r^{K+1-t} \geq L^K + 2L^K - 2L^K = \Omega(L^K)$$

Similar to the RLVR model, we have that by symmetry, it holds that we will have on expectation of W branch entries before reaching the desired target branch. Therefore, the total inference-time compute needed for the SFT model is $\Omega(WL^K)$, and the desired result follows. □

E Distilling RLVR Reasoning Traces

In this section, we will prove that fine-tuning our pretrained model on the reasoning traces of the RLVR-tuned model will also be effective in mitigating the inefficiency caused by a model trained only on golden examples. We will once again invoke Lemma 5.

Proof of Theorem 5. Let $\mathcal{Q}_{\text{distill}}$ be the distribution over adjacent transition pairs (s, a) induced by $(x, y) \sim \mathcal{D}$; that is, sample (x, y) and then sample a uniformly random transition (y_{t-1}, y_t) from the trace. Then

$$L_{\text{distill}}(\Theta) = \mathbb{E}_{(s,a)\sim\mathcal{Q}_{\text{distill}}}\big[-\log \pi_\Theta(a \mid s)\big],$$

and Lemma 5 applies row-wise.

First, note that the distillation target conditionals equal the teacher's policy. Because traces are generated on-policy from $\pi^\star$ (and the policy is assumed not to depend on x beyond the stopping rule), for any state s that occurs before termination we have

$$p_{\mathcal{Q}_{\text{distill}}}(a \mid s) = \pi^\star(a \mid s).$$

Therefore the row-wise cross-entropy for a visited state s is minimized (in policy space) exactly by matching the teacher distribution $\pi^\star(\cdot \mid s)$.

By the support assumption in the theorem statement, each such state s satisfies $d_{\mathcal{Q}_{\text{distill}}}(s) > 0$, so the row is updated by gradient flow. Moreover, in each case the action space out of s splits into two symmetry classes: the desired action(s) and the undesired action(s). With the pretrained initialization, logits are equal within each symmetry class, and the same uniqueness-of-ODE argument as in the pretraining/SFT proofs implies that they remain equal within each class over time. Thus each such row reduces to a two-logit system (desired logit $u(t)$, undesired logit $v(t)$), and the total desired probability $p(t)$ (which is one of a_j, b_j, c_j, d_j depending on the state type) takes the form

$$p(t) = \frac{m_1 e^{u(t)}}{m_1 e^{u(t)} + m_0 e^{v(t)}}$$

for some class sizes m_1, m_0 (e.g. $m_1 = 1, m_0 = L$ for R_j^+; $m_1 = L, m_0 = 1$ for L_j^+; etc.).

Because the teacher conditional is supported entirely on the desired class, Lemma 5 gives (up to the positive row weight, which only rescales time) the same monotone gap dynamics as in the SFT proof: the logit gap $g(t) := u(t) - v(t)$ satisfies $g'(t) \geq 0$ whenever $p(t) < 1$, and in fact the exponential gap $w(t) := e^{g(t)}$ grows at least linearly in t. Consequently $1 - p(t)$ decays to 0, and for any $\kappa > 0$ there exists finite time $T_s(\kappa)$ such that $p(t) \geq 1 - \kappa$ for all $t \geq T_s(\kappa)$.

Applying this argument to every visited state-type $R_j^+, L_j^+, R_j^-, L_j^-$ and taking $T(\kappa) := \max_s T_s(\kappa)$ over these finitely many state types yields

$$\min_j \min\{a_j(t), b_j(t), c_j(t), d_j(t)\} \geq 1 - \kappa \quad \text{for all } t \geq T(\kappa),$$

as desired.

Once a_j, b_j, c_j, d_j are all arbitrarily close to 1, the distilled policy behaves the same as $\pi^\star$ on the branch-dynamics that control forward progress and backtracking, so it inherits the same $\Theta(WK)$ expected hitting-time bound established in Section D for the RLVR policy in the main-text efficiency theorem. □