Neglected Free Lunch from Post-training: Progress Advantage for LLM Agents

Anonymous Author(s)
Affiliation
Address
email

Abstract

Process reward models enable fine-grained, step-level evaluation of LLMs, yet building them for agentic settings remains prohibitively difficult: long-horizon interactions, irreversible actions, and stochastic environment feedback make both human annotation and Monte Carlo estimation infeasible at scale. In this work, we show that reinforcement learning (RL) post-training already provides the ingredients for effective step-level scoring, eliminating the need for dedicated reward model training altogether. Concretely, we derive an implicit advantage under a general stochastic Markov decision process, which we term *progress advantage*—log-probability ratio between the RL-trained policy and its reference policy exactly recovers the optimal advantage function. This formulation makes the resulting signal annotation-free, domain-agnostic, and available as a byproduct of the standard RL post-training pipeline. We validate the effectiveness of the progress advantage across three different applications: *test-time scaling*, *uncertainty quantification*, and *failure attribution* on five benchmarks and four model families. Across all settings, it consistently outperforms confidence-based baselines and, despite requiring no task-specific training, surpasses dedicated trained reward models. We complement these results with deeper analyses on characteristics of progress advantage, offering practical guidance for adoption in real-world agentic systems. Codebase: 🔗

1 Introduction

Reinforcement learning (RL) has become the dominant paradigm for post-training large language models (LLMs), producing agents that can operate autonomously across complex, multi-turn tasks involving tool use, web navigation, and code execution [1, 2, 3, 4]. A central challenge in deploying these agents is evaluating the quality of their behavior, so-called *reward*, at the level of individual steps rather than only at the end of a trajectory. Outcome reward models assign a single scalar to a generated output [5, 6, 7, 8, 9], but this coarse signal provides little guidance for credit assignment over trajectories that may span hundreds of actions. Process reward models (PRMs) address this by providing step-level supervision [10, 11, 12, 13, 14], enabling finer-grained trajectory evaluation that benefits test-time scaling, runtime monitoring, and failure diagnosis. However, PRMs have been explored mostly in mathematical reasoning, and they remain largely underexplored for LLM agents.

Unfortunately, building PRMs for LLM agents is notoriously difficult: agentic trajectories span long horizons and pass through stateful environments where actions such as sending an email or deleting a file are irreversible, preventing the backtracking and repeated rollouts that traditional Monte Carlo estimation relies on [11, 13]. Collecting step-level human annotations in this setup is prohibitively expensive, and even when domain-specific (process) reward models can be trained, they often fail to generalize across tasks or environments [15, 16, 17, 18]. The result is a conspicuous gap: the agents that most need process-level evaluation are precisely the ones for which building process reward models is least feasible.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

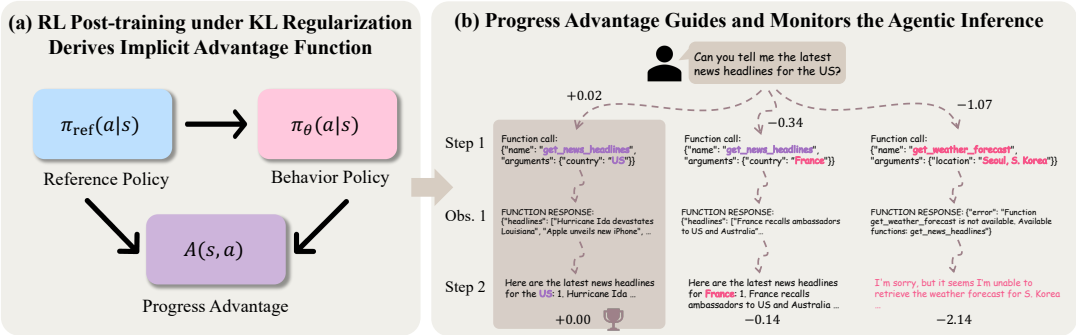

Figure 1: **Framework overview**. (a) We derive an optimal advantage function from an RL-trained policy and its reference policy, which can (b) score the LLM agent trajectories at both the step and trajectory levels without dedicated reward model training.

In this paper, we take a fundamentally different approach. Rather than collecting process annotations or training dedicated reward models [10, 11, 19, 20, 21, 22], we show that RL post-training already freely encodes a process-level signal that can be directly used for inference-time scoring. Concretely, the log-probability ratio between the trained policy and its reference policy—readily available from standard RL post-training—constitutes a theoretically grounded measure of per-step progress, which we term **progress advantage**. We prove that progress advantage exactly recovers the optimal advantage function under the general stochastic MDP (Proposition 1). While prior implicit PRM approaches [23, 24] exploit similar likelihood-based signals in deterministic reasoning settings, agentic environments involve stochastic transitions and external interactions, where such interpretations no longer directly apply (Remark 1). We show that progress advantage naturally corresponds to an advantage function for assessing an agent's sequence of actions in this setting, providing a principled and practical signal for process-level evaluation.

Notably, progress advantage has several appealing properties. It is *annotation-free* and computed from checkpoint pairs that already exist as artifacts of post-training. It is *general* and valid for most of mainstream RL algorithms, including those with explicit KL penalties such as GRPO [8] and those with only clipping-based surrogates such as DAPO [25] (Proposition 2). It is *domain-agnostic* because it emerges from the general post-training phase rather than the task-specific adaptation stage, transferring across tasks without needs for retraining. See Figure 1 for an illustrative overview.

We extensively validate progress advantage across three inference-time applications on multiple agent benchmarks (BFCLv4-MT [26], WebShop [27], AgentDojo [28], τ^2-bench [29], and Who & When [30]) and four model families (Gemma4 [31], Qwen3.5 [32], Qwen3 [33], and Olmo3 [34]). In test-time scaling, progress advantage scores best-of-N trajectory candidates to boost task success rates, outperforming confidence-based baselines, pre-trained reward models, and even task-specific PRMs (Sec. 4.2). In uncertainty quantification, it predicts trajectory-level success or failure with substantially higher AUROC than all baselines, including pre-trained PRMs or a powerful proprietary LLM-as-a-Judge baseline (Sec. 4.3). In failure attribution, it localizes the error step in multi-agent systems, approaching the step-level prediction accuracy of a method specifically trained for this task (Sec. 4.4). These results hold consistently across model families and benchmarks, suggesting that the signal captured by progress advantage is robust and broadly useful.

Contribution: (1) We establish the foundation of implicit reward formulation in stochastic environment and derive *progress advantage* for LLM agents trained by a broad class of RL algorithms; **(2)** We demonstrate its effectiveness across three practical inference-time applications (test-time scaling, uncertainty quantification, and failure attribution), where it outperforms pre-trained reward models without any task-specific training; **(3)** We provide analyses characterizing how progress advantage works, offering practical guidance and insights for real-world adoption.

2 Preliminary

Problem setup and notation. We model the stochastic agents operating in the multi-turn interaction settings as a general token-level Markov Decision Process (MDP), specified by a tuple $(\mathcal{S}, \mathcal{A}, f, r, \rho)$. The state space $\mathcal{S}$ contains states s_t representing the full sequence of tokens generated and observed up to time t, *i.e.*, $s_t = (s_0, a_0, ..., s_{t-1}, a_{t-1})$. The initial state $s_0 \sim \rho$ corresponds to the input prompt,

such as a task specification or a user's initial query to the agent, sampled from the prompt distribution ρ. The action $a_t \in \mathcal{A}$ denotes the token generated at step t by the agent policy $a_t \sim \pi(\cdot|s_t)$. The state transition dynamics is shaped by function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which is a stochastic transition, $s_{t+1} \sim f(\cdot|s_t, a_t)$ with a valid probability distribution ($\sum f(s_{t+1}|s_t, a_t) = 1$) if a_t is the end of sequence token (`EOS`) for each step. Otherwise, it is a concatenation $s_{t+1} = f(s_t, a_t) = [s_t; a_t]$. The stochastic transition brings external observations, such as user messages or tool-calling outputs, into the agent's context. Finally, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ produces a scalar reward per token. The objective is to find a policy that maximizes the expected cumulative rewards.

RL fine-tuning of LLMs. Beyond imitation learning from expert trajectories through supervised fine-tuning (SFT), modern LLMs undergo a reinforcement learning phase, as a central component of post-training. Several popular algorithms have been proposed to realize this phase, including PPO [35] and GRPO [8]. Despite varying in their advantage estimation techniques and surrogate objectives, these methods share a common abstraction, which is a KL-regularized reward maximization problem:

$$\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} \left[\sum_t r(s_t, a_t) - \beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} | s_0 \sim \rho \right], \tag{1}$$

where $\beta > 0$ is the regularization coefficient and π_{ref} is a reference policy, typically built with pre-trained or SFT checkpoints.

Reward reparameterization in deterministic MDP. The objective in Eq. 1 admits an equivalent formulation as a maximum-entropy RL problem, which has a fixed point solution [36], $\pi^*(a_t|s_t) = \exp\left(\frac{Q^*(s_t,a_t) - V^*(s_t)}{\beta}\right)$, where $Q^*(s, a)$ is the optimal action-value function, representing the expected cumulative future reward from the state-action pair under π^*, and $V^*(s) = \mathbb{E}_{\pi^*}[Q^*(s, a)]$ denotes the optimal value function under π^*. The *Bellman equation* relates these terms recursively:

$$Q^*(s_t, a_t) = \begin{cases} r(s_t, a_t) + \beta \log \pi_{\text{ref}}(a_t|s_t) + \mathbb{E}_{s_{t+1} \sim f}[V^*(s_{t+1})], & \text{if } s_{t+1} \text{ is not terminal} \\ r(s_t, a_t) + \beta \log \pi_{\text{ref}}(a_t|s_t), & \text{otherwise.} \end{cases} \tag{2}$$

When the MDP is deterministic, prior work [23, 24] showed that $\mathbb{E}[V^*(s_{t+1})]$ reduces to $V^*(s_{t+1})$, resulting in a clean form of implicit reward,

$$r(s_t, a_t) := \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}, \tag{3}$$

where $V^*(s_T) = 0$ for all terminal states. However, the deterministic MDP assumption—while sufficient for text completion-based non-interactive reasoning tasks—*breaks down in multi-turn stochastic agent settings*, where external observations such as user responses, tool outputs, and environment feedback introduce non-deterministic transitions. Beyond this theoretical gap, there is also a utilization gap. Prior work has primarily leveraged implicit rewards as a training-time objective for policy optimization [23, 37, 38, 39, 40], yet their potential as inference-time scoring signals remains underexplored.

3 Implicit Process Reward Modeling for LLM Agents

Building PRMs for agents is notoriously difficult due to the long-horizon, stateful nature of the agentic environment, making step-level annotation prohibitively expensive, and Monte Carlo estimation—the standard workaround in reasoning tasks [11, 13]—becomes infeasible. Moreover, PRMs trained for a specific task often exhibit poor generalization across different experimental setups [15, 16]. Here, we take a fundamentally new approach. Rather than collecting process annotations or training dedicated PRMs, we show that general RL post-training already yields effective per-token scores at inference time, which aggregate into reliable process-level signals for free.

3.1 Implicit Rewards Under Stochastic Transitions in Agent Systems

We begin by revisiting the implicit reward formulation in Eq. 3. An appealing property of this result is that it holds for *any* policy satisfying the KL-regularized RL fixed-point condition, including models that have already undergone standard post-training. That is, given an RL-trained checkpoint π^* and its reference π_{ref} (e.g., `Qwen3.5-9B` and `Qwen3.5-9B-Base`), one can readily compute per-token implicit rewards as $\log \pi^*(a_t|s_t) - \log \pi_{\text{ref}}(a_t|s_t)$, without any reward modeling. This quantity approximates the underlying reward that was maximized during RL fine-tuning. Although this reward

reparameterization has been widely adopted to guide RL fine-tuning, no existing works have studied its potential in plug-and-play post-development scenarios for LLM agents that we will explore.

Can we justify the use of the same formulation as an inference-time process reward for agents? Unfortunately, no. The clean cancellation in Eq. 3 relies on deterministic transitions, which fail to hold whenever the agent receives stochastic observations. Under a stochastic transition map $f(\cdot|s_t, a_t)$, the implicit reward acquires additional value function terms that do not cancel as below.

Remark 1 (**Implicit Reward in Stochastic MDP**). *The closed-form solution of KL-regularized RL and the Bellman equation under the MDP with a stochastic transition map $f(\cdot|s_t, a_t)$ results in the following token-level reward for $t = 0, ..., T-1$ and trajectory-level reward,*

$$r(s_t, a_t) = \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + V^*(s_t) - \mathbb{E}_{s_{t+1}\sim f(\cdot|s_t,a_t)}[V^*(s_{t+1})], \tag{4}$$

$$\sum_{t=0}^{T-1} r(s_t, a_t) = \sum_{t=0}^{T-1} \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + \sum_{t=0}^{T-1} \delta_t, \tag{5}$$

where $\delta_t = V^*(s_t) - \mathbb{E}_{s_{t+1}\sim f(\cdot|s_t,a_t)}[V^*(s_{t+1})]$ and T denotes the trajectory length.

Please refer to the Appendix D for the derivation. The residual terms δ_t in Eq. 5 capture the discrepancy between the value of the current state and the expected value of the next state under the stochastic transition, which vanishes through telescoping sum when transitions are deterministic. Since V^* is not directly accessible from the policy pair alone, the log-probability ratio no longer recovers the exact reward. This raises a natural question: if the exact reward is out of reach with only the given policy distributions, can we still extract a useful process-level signal in a stochastic MDP?

3.2 Progress Advantage for Agents in Stochastic World

Although the exact reward is irrecoverable from policy distributions alone, we show that a closely related and practically sufficient signal *is* directly derivable. The key insight is to shift the goal: instead of recovering the absolute reward $r(s_t, a_t)$, we target the advantage function $A(s_t, a_t)$: the relative merit of taking actions a_t compared to the average action under the optimal policy at state s_t. This turns out to have a remarkably clean form, as stated below.

Proposition 1 (**Progress Advantage in Stochastic MDP**). *Let $\tilde{\pi}^*$ be an optimal policy under the KL-regularized RL objective (Eq. 1) with $\beta > 0$, shaped with the reference policy π_{ref} where $\pi_{\text{ref}}(a|s) > 0$ for any $a \in \mathcal{A}$ and $s \in \mathcal{S}$. Then, the optimal advantage function is exactly recovered by the log probability ratio between $\tilde{\pi}^*$ and π_{ref} for any state and action:*

$$\tilde{A}^*(s, a) = \tilde{Q}^*(s, a) - \tilde{V}^*(s) = \beta \log \frac{\tilde{\pi}^*(a|s)}{\pi_{\text{ref}}(a|s)}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{6}$$

The proof (Appendix E.1) follows from the Lagrangian solution of the per-state optimization induced by the soft Bellman equation. Critically, while environment stochasticity complicates exact reward recovery (Remark 1, the advantage function we derived natively isolates this by definition. Rather than canceling out the stochastic algebraically, the log probability ratio term naturally absorbs the expected future values, allowing us to extract the exact advantage $Q^*(s, a) - V^*(s)$ without requiring knowledge of the transition model.

We term this quantity **progress advantage**: it measures the expected return of taking a specific action a_t relative to the average action at state s_t, providing a useful, fine-grained signal of whether the agent is making progress toward task completion. Fundamentally, the advantage function is the canonical quantity that drives policy improvement in reinforcement learning. The policy gradient theorem [41] establishes that the gradient of the expected return decomposes as $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[A^{\pi_\theta}(s, a)\nabla_\theta \log \pi_\theta(a|s)]$—it is the advantage that determines which actions should be reinforced and which should be suppressed. Analogously, at inference time, the advantage serves as a practical state-normalized statistic for action evaluation: it tells us exactly how much better or worse a specific action is than the learned policy's expected actions, with all shared context already factored out. While the reward conflates two signals (the inherent difficulty of the state and the quality of the action taken in it), the advantage can isolate the latter. A high reward at an easy state and a moderate reward at a hard state may reflect identical action quality; but the advantage disentangles the two. For scoring agent trajectories at inference time, this disentanglement is useful for comparing across different steps within a trajectory and across trajectories facing different environmental conditions.

Generality beyond explicit KL regularization. A natural concern is whether Proposition 1 applies only to methods with an explicit KL penalty (e.g., PPO [35] and GRPO [8] with adaptive KL). In practice, many widely adopted algorithms (e.g., DAPO [25] and Dr. GRPO [42]) instead use a clipping-based surrogate objective. We show that these methods are also covered:

Proposition 2 (Implicit KL Constraint of Clipping Surrogate RL). *Let π_{ref} and π_θ be the reference and target policies sharing the same support. Define the importance sampling ratio as $R(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)}$. If optimization enforces a per-sample constraint $R(s,a) \in [1-\varepsilon, 1+\varepsilon]$ for all (s,a) and a small $\varepsilon > 0$, then $D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) \lesssim \frac{\varepsilon^2}{2}$, similarly for reverse KL, locally at $R(s,a) \approx 1$.*

Proposition 2 (proof in Appendix E.2) establishes that clipping-based surrogate objective [35] derives a conservative KL-constrained solution policy governed by the small clipping threshold ϵ.

> **Takeaways.** We derived progress advantage in stochastic MDP, which is computed from the optimal behavior policy and reference policy pair (Proposition 1) to measure the relative merit of each action for scoring agent trajectories. The progress advantage is valid for the broad class of policy trained via RL objectives with regularization, whether they contain an explicit KL regularization or a clipping-based surrogate (Proposition 2).

3.3 From Theory to Practical Implementation

Translating progress advantage into practice requires three design decisions: specifying policies, aggregating per-token advantages into process-level scores, and representing the token probability.

Policy specification. For $\tilde{\pi}^*$, one can use any policy models trained via a KL-regularized or a clipping-based RL objective, covering virtually most mainstream post-training pipelines in use today. Meanwhile, the selection of the reference policy π_{ref} depends on the RL pipeline adopted to get $\tilde{\pi}^*$: it can be a pre-trained base checkpoint in RL-Zero settings [9], an SFT checkpoint in standard single-stage RL, or a previous round's policy in online iterative RL [43]. The key consideration is that π_{ref} should be neither too distant nor too close to $\tilde{\pi}^*$. If too far, the log-ratio is dominated by generic distributional differences rather than task-relevant distinctions; if too close, the signal is insufficient to distinguish between good and poor actions. Since most model providers do not release their intermediate checkpoints, we confine our analysis to publicly available policy pairs (see Appendix B for the full list). In Figure 5 of Sec. 4.5, we further empirically analyze how the choice of reference policy affects the utility of the progress advantage.

Progress advantage aggregation. Since Eq. 6 produces a token-level advantage $\tilde{A}^*(s_t, a_t)$ at each position t, we need aggregation strategies to obtain step-level and trajectory-level scores suited to each application. Table 1 shows some natural choices. Simple summation yields the standard additive trajectory advantage, while averaging produces a length-normalized variant that prevents long trajectories from being scored higher. In addition, one can inject an inductive bias based on their knowledge to implement position-weighted advantage. Extreme token advantage (min or max) captures the worst or best-case token advantage within a sub-trajectory. The aggregation choice can meaningfully affect the quality of progress advantage (Figure 4), and we pick the best per task.

Table 1: Sub-trajectory advantage aggregation over a set of consecutive token indices $\mathcal{I}$.

Aggregation	Interpretation
$\sum_{t\in\mathcal{I}} \tilde{A}^*(s_t, a_t)$	Vanilla sub-trajectory advantage
$\frac{1}{\lvert\mathcal{I}\rvert} \sum_{t\in\mathcal{I}} \tilde{A}^*(s_t, a_t)$	Per-token average advantage
$\sum_{t\in\mathcal{I}} w_t \tilde{A}^*(s_t, a_t)$	Position-weighted advantage
$\min_{t\in\mathcal{I}} \log \tilde{\pi}^* - \min_{t\in\mathcal{I}} \log \pi_{\text{ref}}$ (resp. max)	Extreme token advantage

Representing the token probability. The clean implementation of Eq. 6 is directly using $\tilde{\pi}^*(a_t|s_t)$ (and $\pi_{\text{ref}}(a_t|s_t)$), but the pure token probability is often noisy and a source of instability during RL [44, 45]. Thus, we also explore a top-k average token probability variant in Appendix A and C.

4 Empirical Validation

We evaluate progress advantage on three inference-time applications that collectively test whether the signal is useful for (1) parallel test-time scaling through best-of-N sampling, (2) uncertainty quantification, and (3) failure attribution. Appendix B and C cover additional details and results.

Table 2: **Test-time scaling through best-of-8 sampling**. We compare reward scoring methods across four benchmarks and two LLM backbones, reporting the success rate (%) of the selected trajectory. Our progress advantages successfully boost the success rate, especially when the non-zero-temperature exploratory behavior is beneficial to the task, i.e., WebShop and τ^2-Airline.

Scoring Method	Training	BFCLv4-MT		WebShop		AgentDojo		τ^2-Airline		Average	
		`Gemma4-4B`	`Qwen3.5-9B`	`Gemma4-4B`	`Qwen3.5-9B`	`Gemma4-4B`	`Qwen3.5-9B`	`Gemma4-4B`	`Qwen3.5-9B`	`Gemma4-4B`	`Qwen3.5-9B`
Pass@N (oracle)	✗	22.0	48.0	53.0	49.0	48.5	94.8	58.0	78.0	45.4	67.5
Greedy Decoding	✗	19.0	42.5	32.0	21.0	48.5	94.8	34.0	60.0	33.4	54.6
Mean-of-N	✗	17.5	38.5	41.6	26.4	38.8	89.0	34.5	64.8	33.1	54.7
WildReward-8B [46]	✓	**20.0**	**42.5**	41.0	26.0	43.3	86.6	28.0	64.0	33.1	54.8
ThinkPRM-7B [47]	✓	18.5	37.0	38.0	22.0	37.1	85.6	30.0	64.0	30.9	52.2
ThinkPRM-14B [47]	✓	19.0	40.0	43.0	33.0	**44.3**	88.7	28.0	58.0	33.6	54.9
Self-Certainty [48]	✗	15.0	34.5	34.0	22.0	33.0	85.6	34.0	64.0	29.0	51.5
DeepConf Tail [49]	✗	15.5	36.0	39.0	28.0	35.1	86.6	36.0	62.0	31.4	53.2
DeepConf B10 [49]	✗	15.5	34.5	35.0	30.0	30.9	86.6	28.0	**72.0**	27.4	55.8
Progress Advantage	✗	19.0	**42.5**	**45.0**	**42.0**	43.3	**91.8**	**48.0**	**72.0**	**38.8**	**62.1**

4.1 Setup

Benchmarks. We ground our evaluation in four benchmarks that represent realistic agentic workloads: BFCLv4-MT [26] (multi-turn tool calling), WebShop [27] (tool-augmented online shopping), AgentDojo [28] (tool-augmented general task solving), and τ^2-bench [50, 29] (conversational agents in customer-service environments). All four require multi-turn interaction with external tools in stateful environments, exercising precisely the stochastic MDP structure that motivates our approach. Each application defines a distinct evaluation protocol. For test-time scaling, we perform best-of-N sampling by generating 8 trajectories per task, with temperature 0.7 for τ^2-bench and WebShop, and 0.4 for the remaining. We score these trajectories with each reward method and measure the average task success rate of the selected trajectories. For uncertainty quantification, we use trajectory-level reward to predict whether each trajectory succeeds or fails on τ^2-bench, measured by AUROC [51]. Failure attribution serves a step-level failure detection, which is described separately in Sec. 4.4.

Models. We mainly evaluate four public LLM families: `Gemma4-4B` [31], `Qwen3.5-9B` [32], `Qwen3-14B` [33], and `Olmo3-7B` [34]. For each, we pair the RL-trained final checkpoint with its corresponding base/intermediate checkpoint as the reference policy to build progress advantage.

Baseline method. We compare against two categories of baselines. *Trained reward models*: (1) WildReward-8B [46], (2) ThinkPRM-7B/14B [47], which are specifically trained on real-world user-agent interactions or multi-step reasoning datasets, and (3) AgentPRM [20] that is specifically trained on a downstream task. *Confidence-based methods*: (4) Self-Certainty [48], which scores trajectories by the average token probability certainty, and (5) DeepConf [49], which proposes step-level confidence aggregation strategies including tail-step confidence and bottom-10% average of step confidences. Crucially, progress advantage and the confidence baselines require no dedicated training, whereas trained reward models usually require task-specific or domain-specific supervision.

4.2 Test-time Scaling

We start with best-of-8 sampling scenarios in Tab. 2, where Pass@N denotes the pass rate of at least one of trajectories, Mean-of-N denotes the mean success rate of them, and Greedy Decoding is a zero-temperature deterministic generation. Overall, progress advantage shows stable performance across datasets and models, outperforming the expensive training-based method and confidence-based methods with significant margins (15.5% for `Gemma4` and 11.3% for `Qwen3.5`) on average. Notably, it consistently beats the baseline methods in cases where high-temperature exploratory trajectories are beneficial over greedy ones, e.g., WebShop and τ^2-Airline. We hypothesize that optimal advantage signals favor setups where both the average (Mean-of-N) and ceiling (Pass@N) values of trajectories are high, connected to the theorem. Table 12 in Appendix C further shows that it even outperforms AgentPRM-7B, specifically trained on a downstream task.

4.3 Uncertainty Quantification

One of the key building blocks for a reliable agent in the wild is a framework for uncertainty quantification. As noted by Oh et al. [51], quantifying uncertainty in a multi-turn interactive inference setup brings non-trivial open problems hard to tackle with existing uncertainty methods. This subsection explores the application of the progress advantage for the UQ of LLM agents. Specifically, we predict whether a trajectory generated by an agent ends with success or not by adopting the

Table 3: **Uncertainty quantification for trajectory monitoring**. We compare scoring methods across four LLM backbones on τ^2-bench Airline and Retail domains to predict an agent's success on each model's greedy-decoding trajectory with trajectory-level scoring, measured by AUROC.

Scoring Method	Training	τ^2-Airline				τ^2-Retail			
		`Gemma4-4B`	`Qwen3.5-9B`	`Qwen3-14B`	`Olmo3-7B`	`Gemma4-4B`	`Qwen3.5-9B`	`Qwen3-14B`	`Olmo3-7B`
Sonnet-4.6 [52]	✗	0.615	0.726	0.519	0.715	0.852	0.899	0.864	0.656
WildReward-8B [46]	✓	0.312	0.540	0.314	0.514	0.643	0.468	**0.689**	0.584
ThinkPRM-7B [47]	✓	0.478	0.582	0.276	0.492	0.469	0.551	0.543	**0.670**
ThinkPRM-14B [47]	✓	0.426	0.655	0.292	0.708	0.573	0.610	0.637	0.544
Self-Certainty [48]	✗	0.840	0.642	0.663	0.486	0.397	0.366	0.608	0.392
DeepConf Tail [49]	✗	0.581	0.588	0.682	0.472	0.382	0.344	0.380	0.608
DeepConf B10 [49]	✗	0.834	0.587	0.636	0.618	0.416	0.496	0.582	0.288
Progress Advantage	✗	**0.865**	**0.720**	**0.739**	**0.799**	**0.690**	**0.678**	0.650	0.664

trajectory-level reward as a (un)certainty signal. Table 3 shows AUROC computed over the whole trajectory samples on τ^2-bench (50 for Airline, 114 for Retail) across four different models, where we score the trajectory generated by each behavior model through its own log probability with (ours) and without (Self-Certainty and DeepConf) reference policy's log probability offset, or a different, pre-trained reward model. We see that the progress advantage remarkably outperforms all the baselines in τ^2-Airline and also shows competitive results on τ^2-Retail, demonstrating its validity under a stochastic MDP under the complex interaction scaffolding (See Figure 9 for more analyses).

We further test whether the progress advantage can predict the chance of success over a trajectory generated by another policy backbone model. In Table 4, we use `Gemma4-4B` as a reward model to score trajectories of `Qwen3.5-9B` and `Qwen3-14B` on τ^2-Airline. We see that the progress advantage acts as an external scorer to assess the quality of the action sequence yielded by a different policy, implying its potential as an off-the-shelf reward model to monitor arbitrary trajectories.

Table 4: **Uncertainty quantification on trajectories generated by a different policy**. `Gemma4-4B` as a reward model scoring trajectories produced by different behavior policies on τ^2-Airline. Trajectory-level AUROC over success prediction (higher is better).

Scoring Method	`Qwen3.5-9B`	`Qwen3-14B`
Self-Certainty [48]	0.587	0.648
DeepConf Tail [49]	0.482	0.610
DeepConf B10 [49]	0.563	0.636
Progress Advantage	**0.754**	**0.727**

4.4 Failure Attribution

An emerging field of agentic system monitoring is *failure attribution*, where we detect a step when the system would make the critical error across the whole trajectory. We evaluate PRMs on Who & When benchmark [30], predicting the decisive error step, t_{err}, over pre-extracted trajectories from multi-agent systems. Here, we make a prediction as an index of the minimum per-step reward, and compare the prediction with the ground truth, i.e., $\mathbb{I}(\arg\min_t \hat{r}_t = t_{\text{err}})$. As shown in Figure 2, the task is notably challenging for pre-trained reward models and even for a task-specific RL-trained baseline, AgenTracer [53]. Our method shows promising results on both splits, while rivaling AgenTracer in the Hand-Crafted split, demonstrating its reliable step-level credit assignment under a carefully designed agentic harness.

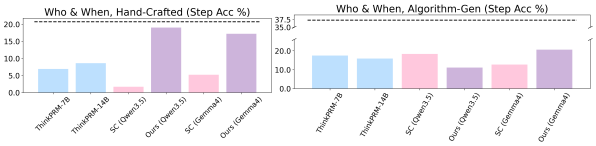

Figure 2: **Who & When step-level accuracy**. We predict when the agent system makes a decisive error. SC denotes Self-Certainty [48], and the dashed line denotes AgenTracer [53], which is specifically trained on this.

4.5 Additional Empirical Study on Progress Advantage

Contrasting $\tilde{\pi}^*$ and π_{ref} is better than sole. Since the progress advantage is defined with two policies, one may wonder if we can just use one of them as a reward with the same aggregation strategies. Tab. 5 provides the average rank of AUROC on τ^2-Airline uncertainty quantification (See Tab. 11 for setup and more results), confirming that progress advantage provides more reliable signals sharpened by contrasting distributions [54]. We go into this in Fig. 3.

Table 5: **Progress advantage and its ingredients on UQ.**

Method	Avg. Rank
Ours	**1.44**
$\log \tilde{\pi}^*(a\|s)$	2.25
$\log \pi_{\text{ref}}(a\|s)$	2.31

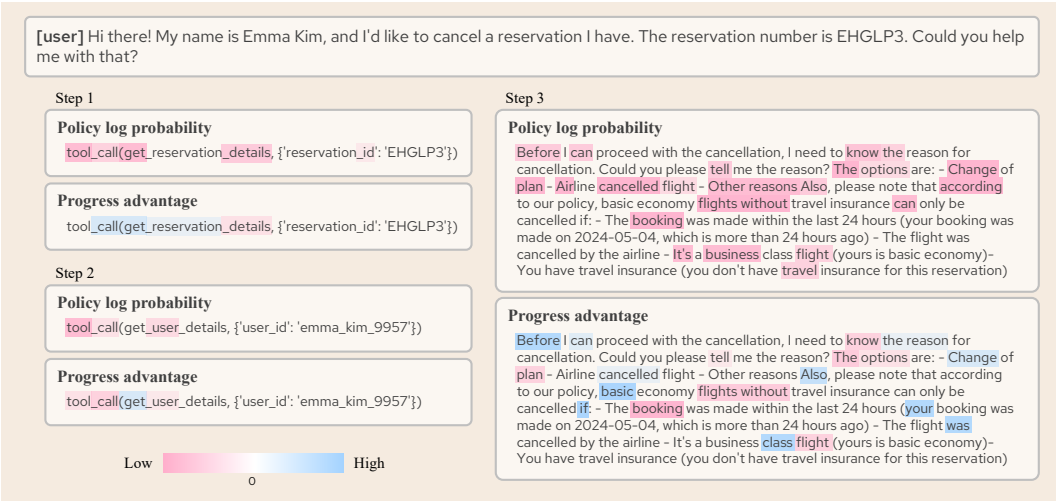

Figure 3: **Qualitative analysis on token-level signals.** Progress advantage effectively rewards actions specifically helpful to achieve the downstream goal, whereas the policy log probability does not.

Per-token qualitative analysis. We perform fine-grained analysis to investigate whether the progress advantage produces reasonable signals related to goal achievement. Figure 3 presents a case in which the agent correctly refuses a reservation cancellation request in conflict with the domain-specific constraint in τ^2-Airline. The results reveal a clear contrast between pure policy log probability $\log \tilde{\pi}^*(\cdot|\cdot)$ and progress advantage $\log \frac{\tilde{\pi}^*(\cdot|\cdot)}{\pi_{\text{ref}}(\cdot|\cdot)}$. Specifically, the pure policy log probability assigns low scores to tool-calling strings (steps 1 and 2) even though they are correct, probably because the frequency of the tool string is lower than plain natural language. In contrast, progress advantage assigns positive scores to these strings, thanks to the offset effect of the reference log probability. Furthermore, the policy log probability penalizes domain constraint-related terms in step 3, such as "change of plan" and "business class" which cover the key criteria of canceling a flight. Progress advantage, on the other hand, rewards most of these terms, demonstrating its awareness of goal-specific information to effectively reward actions that induce success on the task.

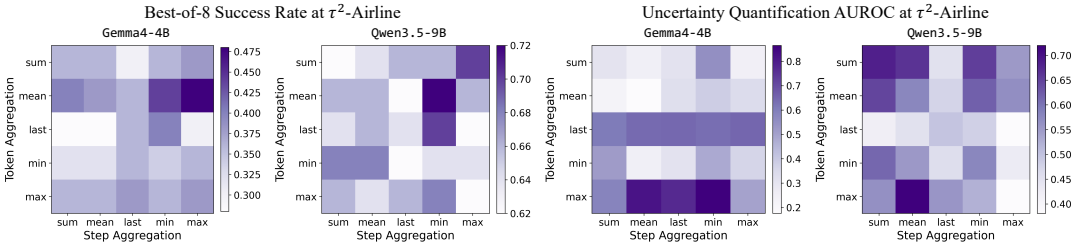

Figure 4: **Combinations of token and step aggregation strategy for progress advantage.** The aggregation across token and step advantages affects the effectiveness of progress advantage, and each downstream task and model shows quite a different flavor in the aggregation strategy.

Advantage aggregation strategy. Since our derived progress advantage (Eq. 1) serves as a token-level signal, we explore aggregation strategies at both token and step levels, following prior work on sentence-level and trajectory-level scores [55, 56, 49]. As shown in Figure 4, different aggregation combinations excel at different applications: the (MEAN, MIN) pair performs strongly for best-of-N selection, while the (MAX, MEAN) pair becomes the winner for UQ. One possible explanation is that step-level MIN aggregation penalizes trajectories containing a low-quality step, thereby favoring trajectories whose progress signal remains consistently positive across the interaction. This can be ensure a better sequence of actions during the negotiation-centric airline tasks. For UQ, on the other hand, focusing on the extrema, such as maximum token advantage, can be a better indicator for per-step success, which may capture salient local evidence crucial for ultimate success. Then, the conventional mean operation over per-step maximum advantages can reliably stand for a per-trajectory uncertainty. This is aligned with findings from the reasoning model inference, where some important tokens drive the final success [57, 58]. Extended results are provided in Figure 7 and 8.

Specification of reference policy. As noted in Sec. 3.3, progress advantage is constructed with the behavior policy and the reference policy, and the reference policy specification becomes a design choice. Rather than simply adopting the base checkpoint version of the final policy, we test policy merging between the final and base checkpoints, $\theta_\alpha = \alpha\theta_{\text{final}} + (1-\alpha)\theta_{\text{base}}$, to get a spectrum of reference policies π_{θ_α} for $\alpha \in \{0.1, ..., 0.9\}$ to construct $\log \frac{\tilde{\pi}^*(a|s)}{\pi_{\theta_\alpha}(a|s)}$ in Figure 5. Here, we adopt two different merging methods, WISE [59] as the aforementioned simple linear interpolation and TIES [60] as an interference-aware robust merging variant. While naive linear interpolation mostly degrades the quality of progress advantage, the robust merging variant consistently induces better results in most α from 0.2 to 0.7. This supports our hypothesis that the reference should not be too far or too close to the behavior policy $\tilde{\pi}^*$, suggesting the promise of progress advantage coupled with advanced merging methods [61, 62, 63, 64, 65] to craft a sharper reference policy.

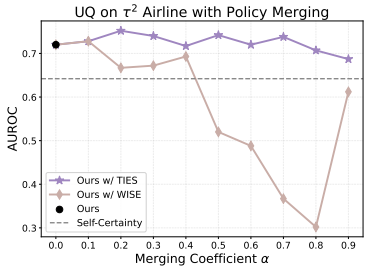

Figure 5: **Varying reference policy.** We merge `Qwen3.5-9B-Base` with `Qwen3.5-9B` in the weight space and use it as π_{ref} in our progress advantage for τ^2-Airline UQ.

5 Related Work

Process Reward Modeling for Reasoning Models. PRMs provide fine-grained supervision over intermediate reasoning steps, assisting models to improve their reasoning quality step-by-step. Early works typically formulate process supervision as a binary classification, where each step is labeled as correct or incorrect [10, 12]. Recent approaches move beyond step-wise classification; instead cast PRM learning as a ranking problem, drawing on Q-value theory [14] or probabilistic formulations of step quality [66]. In parallel, because annotating reasoning trajectories at the step-level is expensive and time-consuming, several works explore efficient or automated strategies to construct step supervision [13, 11, 67]. Notably, implicit PRMs [21] eliminate the need for explicit step annotations during training has been shown to be effective for both test-time scaling and RL of reasoning models [68]. However, these methods assume deterministic and non-interactive text completion for multi-step reasoning; our work differs by deriving an implicit process reward that is theoretically grounded in stochastic MDPs and validated in multi-turn agentic settings.

Process Reward Modeling for LLM Agents. There are emerging research endeavors to build PRMs for LLM agents that use tools and interact with the user and environment to achieve a long-horizon goal. Prior work typically relies on per-step MC estimation [19, 20] to get step-level annotations while confining the scope to relatively simple tasks with a few rounds of turns, which is unreliable [69, 70] and becomes infeasible in complex, long-horizon tasks. Although Liu et al. [22] bypass the process supervision by adopting an implicit reward formulation, they still perform the downstream task-specific training with outcome-level supervision to learn the implicit PRMs. In contrast, we showed that an RL-trained policy, paired with some reference policy, constructs an implicit advantage function that already can be a sufficient signal to guide test-time scaling or post-deployment monitoring, *without any extra training* (See Appendix G for a broader context).

6 Conclusion

LLMs equipped with the agentic harness operate on multi-turn, interactive environments under stochasticity, where measuring the intermediate progress over a goal-oriented trajectory brings huge opportunities; at the same time, the real bottleneck—expensive training of PRMs with process annotations. We establish the theoretical foundation through progress advantage for implicit reward technology in stochastic MDP; offer a novel angle and recipe to build PRMs from LLM checkpoint pairs of RL fine-tuned and its base, bypassing collecting the process labels and dedicated reward model training. Across five agentic benchmarks, four model families, and three downstream applications, progress advantage consistently outperforms confidence-based baselines and matches or beats competitive training-based reward models as well as a proprietary LLM judge. We hope these promising results stimulate a new paradigm of future research towards a scalable and practical approach to process-level guidance and monitoring of real-world agentic systems.

References

[1] David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1:11, 2025.

[2] OpenAI. Chatgpt agent. https://chatgpt.com/features/agent/, 2025. Accessed: 2025-12-11.

[3] Google. Gemini agent. https://gemini.google/overview/agent/, 2025. Accessed: 2025-12-11.

[4] Anthropic. Cowork: Claude code for the rest of your work. https://claude.com/blog/cowork-research-preview, January 2026.

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

[6] Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 858–875, 2024.

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

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

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[10] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

[11] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[12] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[14] Wendi Li and Yixuan Li. Process reward model with q-value rankings. In *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

[16] Liyuan Mao, Haoran Xu, Amy Zhang, Weinan Zhang, and Chenjia Bai. Information-theoretic reward decomposition for generalizable RLHF. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[17] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

[18] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, 2025.

[19] Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

[20] Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, page 4184–4195, 2026.

[21] Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*, 2025.

[22] Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *The Fourteenth International Conference on Learning Representations*, 2026.

[23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[24] Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a q-function. In *First Conference on Language Modeling*, 2024.

[25] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[26] Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[27] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[28] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

[29] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

[30] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 76583–76599. PMLR, 2025.

[31] Google DeepMind. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4, 2026. Last updated April 17, 2026.

[32] Qwen Team. Qwen3.5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

[33] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[34] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

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

[36] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[37] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

[38] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[39] Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, 2024.

[40] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

[41] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 2 edition, 2018.

[42] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025.

[43] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Transactions on Machine Learning Research*, 2024.

[44] Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025.

[45] Penghui Qi, Xiangxin Zhou, Zichen Liu, Tianyu Pang, Chao Du, Min Lin, and Wee Sun Lee. Rethinking the trust region in llm reinforcement learning. *arXiv preprint arXiv:2602.04879*, 2026.

[46] Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Lei Hou, and Juanzi Li. Wildreward: Learning reward models from in-the-wild human interactions. *arXiv preprint arXiv:2602.08829*, 2026.

[47] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

[48] Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[49] Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026.

[50] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

[51] Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Xuefeng Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. Uncertainty quantification in llm agents: Foundations, emerging challenges, and opportunities. *arXiv preprint arXiv:2602.05073*, 2026.

[52] Anthropic. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, February 2026.

[53] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng YAN. Agentracer: Who is inducing failure in the LLM agentic systems? In *The Fourteenth International Conference on Learning Representations*, 2026.

[54] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312, 2023.

[55] Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. Enhancing uncertainty-based hallucination detection with stronger focus. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 915–932, 2023.

[56] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063, 2024.

[57] Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[58] Jaehui Hwang, Dongyoon Han, Sangdoo Yun, and Byeongho Heo. Oops, wait: Token-level signals as a lens into llm reasoning. *arXiv preprint arXiv:2601.17421*, 2026.

[59] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7959–7971, June 2022.

[60] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

[61] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

[62] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[63] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

[64] Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pages 207–223. Springer, 2024.

[65] Changdae Oh, Yixuan Li, Kyungwoo Song, Sangdoo Yun, and Dongyoon Han. Dawin: Training-free dynamic weight interpolation for robust adaptation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[66] Zheng Zhang, Ziwei Shan, Kaitao Song, Yexin Li, and Kan Ren. Linking process to outcome: Conditional reward modeling for LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[67] Nakyung Lee, Sangwoo Hong, and Jungwoo Lee. Efficient process reward modeling via contrastive mutual information. *arXiv preprint arXiv:2604.10660*, 2026.

[68] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[69] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, 2025.

[70] Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. VersaPRM: Multi-domain process reward model via synthetic reasoning data. In *Forty-second International Conference on Machine Learning*, 2025.

[71] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[72] David Hall, Ahmed Ahmed, Christopher Chou, Abhinav Garg, Rohith Kuditipudi, Will Held, Nikil Ravi, Herumb Shandilya, Jason Wang, Jason Bolton, Siddharth Karamcheti, Suhas Kotha, Tony Lee, Nelson Liu, Joel Niklaus, Ashwin Ramaswami, Kamyar Salahi, Kaiyue Wen, Chi Heem Wong, Sherry Yang, Ivan Zhou, and Percy Liang. Introducing Marin: An open lab for building foundation models. Marin Community Blog, May 2025. Blog post.

[73] Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025.

[74] Vedant Shah, Johan Obando-Ceron, Vineet Jain, Brian Bartoldson, Bhavya Kailkhura, Sarthak Mittal, Glen Berseth, Pablo Samuel Castro, Yoshua Bengio, Nikolay Malkin, et al. A comedy of estimators: On kl regularization in rl training of llms. *arXiv preprint arXiv:2512.21852*, 2025.

[75] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[76] Cheng-Han Chiang and Hung-yi Lee. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, 2023.

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

[78] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, 2023.

[79] Yann Dubois, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*, 2024.

[80] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *Forty-first International Conference on Machine Learning*, 2024.

[81] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *International Conference on Machine Learning*, pages 57905–57923. PMLR, 2024.

[82] Guijin Son, Hyunwoo Ko, Hoyoung Lee, Yewon Kim, and Seunghyeok Hong. LLM-as-a-judge & reward model: What they can and cannot do. *arXiv preprint arXiv:2409.11239*, 2024.

[83] Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Forty-second International Conference on Machine Learning*, 2025.

[84] Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters*, 140:325–331, 2020.

[85] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[86] Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of LLMs should not ignore non-determinism. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4195–4206, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[87] Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[88] Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. Agentic uncertainty quantification. *arXiv preprint arXiv:2601.15703*, 2026.

[89] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021.

[90] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[91] Kimi Team. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[92] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024.

[93] Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[94] Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*, 2025.

[95] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang, Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

[96] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[97] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[98] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361. PMLR, 2017.

[99] Ronald A Howard. Dynamic programming and markov processes., 1960.

[100] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[101] Chengqi Lyu, Songyang Gao, Yuzhe Gu, Wenwei Zhang, Jianfei Gao, Kuikun Liu, Ziyi Wang, Shuaibin Li, Qian Zhao, Haian Huang, Weihan Cao, Jiangning Liu, Hongwei Liu, Junnan Liu, Songyang Zhang, Dahua Lin, and Kai Chen. Exploring the limit of outcome reward for learning mathematical reasoning. In *Second Conference on Language Modeling*, 2025.

[102] Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 18806–18827. Curran Associates, Inc., 2023.

[103] Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen. Beyond reverse KL: Generalizing direct preference optimization with diverse divergence constraints. In *The Twelfth International Conference on Learning Representations*, 2024.

[104] Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

[105] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2312–2324, 2024.

[106] BARNET WOOLF. The log likelihood ratio test (the g-test). *Annals of Human Genetics*, 21(4):397–409, 1957.

[107] Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2):307–333, 1989.

[108] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[109] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012.

[110] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3698–3707, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[111] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[112] Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pages 33–40. PMLR, 2005.

[113] Changdae Oh, Heeji Won, Junhyuk So, Taero Kim, Yewon Kim, Hosik Choi, and Kyungwoo Song. Learning fair representation via distributional contrastive disentanglement. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1295–1305, New York, NY, USA, 2022. Association for Computing Machinery.

[114] Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023.

[115] Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting grpo beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25559–25571, 2025.

[116] Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. In *The Fourteenth International Conference on Learning Representations*, 2026.

[117] Colin Raffel. Building machine learning models like open source software. *Communications of the ACM*, 66(2):38–40, 2023.

[118] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022.

[119] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *Advances in Neural Information Processing Systems*, volume 35, pages 29262–29277, 2022.

[120] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023.

[121] Jennifer A Hoeting, David Madigan, Adrian E Raftery, and Chris T Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, 14(4):382–417, 1999.

[122] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[123] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[124] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[125] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017.

[126] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

[127] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, 2025.

[128] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR, 2022.

[129] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, 2022.

[130] Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3201–3219, 2024.

[131] Changdae Oh, Gyeongdeok Seo, Geunyoung Jung, Zhi-Qi Cheng, Hosik Choi, Jiyoung Jung, and Kyungwoo Song. Robust adaptation of foundation models with black-box visual prompting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[132] Jiya Manchanda, Laura Boettcher, Matheus Westphalen, and Jasser Jasser. The open source advantage in large language models (llms). *arXiv preprint arXiv:2412.12004*, 2024.

[133] Dongyoon Han, Junsuk Choe, Seonghyeok Chun, John Joon Young Chung, Minsuk Chang, Sangdoo Yun, Jean Y Song, and Seong Joon Oh. Neglected free lunch-learning image classifiers using annotation byproducts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20200–20212, 2023.

[134] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[135] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

[136] Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. SELF-GUARD: Empower the LLM to safeguard itself. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1648–1668, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

Appendix

Contents

A Implementation of Progress Advantage **19**

A.1 Policy Pairs in Progress Advantage 19

A.2 Progress k-Advantage 19

B Details on Experiment Setup **19**

B.1 Baseline Methods 19

B.2 Application-Specific Details 21

B.3 Inference Setup and Configuration 23

C Additional Results **23**

D Derivation of Implicit Rewards Under Stochastic MDP **27**

E Missing Proof **28**

E.1 Derivation of Progress Advantage 28

E.2 Proof: Clipping Surrogate RL as an Implicit KL Constraint 29

F Limitation and Future Work **30**

G Broader Context and Discussion **30**

G.1 Outcome Reward Modeling 30

G.2 Implicit Reward Modeling 30

G.3 Distribution Contrasting and Sharpening 31

G.4 Open-source AI and Sustainable Machine Learning 31

H Broader Impacts **31**

I Computing Resource Statement **31**

J Prompt Template for Baseline Methods **32**

A Implementation of Progress Advantage

A.1 Policy Pairs in Progress Advantage

Table 6: **Policy pair lineup used for progress advantage construction.** We consider the following five open-source model families, which offer the base or intermediate checkpoint of their final post-trained version.

Model Family	Behavior Policy $\tilde{\pi}^*$	Reference Policy π_{ref}	HuggingFace Collection URL
Qwen3.5 [32]	Qwen3.5-9B	Qwen3.5-9B-Base	https://huggingface.co/collections/Qwen/qwen35
Qwen3 [33]	Qwen3-14B	Qwen3-14B-Base	https://huggingface.co/collections/Qwen/qwen3
Qwen2.5 [71]	Qwen2.5-7B-Instruct	Qwen2.5-7B	https://huggingface.co/collections/Qwen/qwen25
Gemma4 [31]	gemma-4-E4B-it	gemma-4-E4B	https://huggingface.co/collections/google/gemma-4
Olmo3 [34]	Olmo-3-7B-Instruct	Olmo-3-7B-Instruct-DPO	https://huggingface.co/collections/allenai/olmo-3

Progress advantage (Proposition 1) is built upon two policies: the RL-trained behavior policy and its reference policy used as a regularization pivot during RL training. Due to limited public resources in terms of the available policy pairs, we consider five representative model families in Table 6 to evaluate the progress advantage in practice. It is common for the industry to just release the final post-trained model to the public while keeping their base and intermediate model checkpoints confidential, but we call on institutions and communities to pay attention and effort to realize a more transparent model development pipeline with fully opened intermediate artifacts, i.e., checkpoints and datasets, as well as step-by-step recipes as pushed by a few leaders [72, 34, 73].

A.2 Progress k-Advantage

The default implementation of progress advantage is just using the token probability from $\tilde{\pi}^*$ and π_{ref} of the realized token at each position in the trajectory. However, some recent works [44, 74, 45] found that naive token-probability-based regularized RL training sometimes induces nuance instability in gradient estimation; at the same time, confidence-based LLM self-evaluation methods, such as Self-Certainty [48] and DeepConf [49], typically adopt probability smoothing over multiple tokens (top-k for instance) to derive stable implementation of confidence rather than the pure (log) probability. Therefore, we additionally propose progress k-advantage, a top-k smoothed probability variant of standard progress advantage formulated as below,

$$\text{Progress } k\text{-Advantage} := \frac{1}{k}\Big(\sum_{i\in\text{Top-}k} \log \tilde{\pi}^*(A[i] \mid s) - \sum_{i\in\text{Top-}k} \log \pi_{\text{ref}}(A[i] \mid s)\Big), \tag{7}$$

where $\pi(A[i] \mid s)$ denotes the i-th token probability value from the output action probability distribution $\pi(A \mid s)$ of the policy π. In Appendix C, we compare this top-k-smoothed log probability version with the vanilla version.

B Details on Experiment Setup

B.1 Baseline Methods

Self-Certainty [48] is a training-free, self-confidence-based scoring method proposed for best-of-N selection in reasoning tasks. It evaluates trajectory quality using only the behavior policy's output token probability distribution. To be specific, at each position t, it measures the KL divergence between the model's next-token distribution $\tilde{\pi}^*(\cdot \mid s_t)$ and the uniform distribution over the action space $\mathcal{A}$, capturing how peaked the policy is in its predictions. Equivalently, up to an additive constant, it can be written as a cross-entropy with the uniform distribution. The trajectory-level score for a trajectory τ is defined by averaging the per-token self-certainty across all T positions, like below:

$$\text{Self-Certainty}(\tau) \;=\; -\frac{1}{T\,|\mathcal{A}|}\sum_{t=0}^{T-1}\sum_{a\in\mathcal{A}} \log \tilde{\pi}^*(a \mid s_t). \tag{8}$$

We use a top-20 truncated distribution rather than the full vocabulary distribution for practicality. This provides a scalable signal for best-of-N selection with no additional training, but in agentic settings, it rewards fluent continuations regardless of goal progress, sometimes resulting in poor scoring on correct but low-frequency tool-call strings.

DeepConf [49] is also a training-free, confidence-based scoring method designed and validated mainly on math or STEM domain reasoning tasks. It refines confidence estimation by introducing local confidence measures over sliding token windows, motivated by the observation that global trace-level averages can dilute some important finer signals per step. At each position t, the per-token confidence is defined as the average log-probability of the top-k tokens under $\tilde{\pi}^*(\cdot \mid s_t)$ as $C_t = \frac{1}{k} \sum_{a \in \text{Top-}k(\tilde{\pi}^*(\cdot \mid s_t))} \log \tilde{\pi}^*(a \mid s_t)$. The group confidence is then defined over by sliding window $G_i = \{i - w, i - w + 1, ..., i\}$ with w previous tokens, such as, $C_{G_i} = \frac{1}{|G_i|} \sum_{t \in G_i} C_t$. While DeepConf authors adopted an overlapping sliding window to define group confidence, this is not valid for agent rollouts that mix the agent's own output tokens with the environment-side observation tokens. Therefore, we define the group confidence as the average of token confidence per agent's action step without overlapping neighbor steps. Finally, to define a trajectory-level score for the TTS and UQ scenarios, we adopt two instantiations of DeepConf as follows.

(1) *DeepConf Tail* as the last step group confidence, capturing action quality at the termination phase:

$$\text{DeepConf}_{\text{Tail}}(\tau) \;=\; C_{G_{\text{last}}}. \tag{9}$$

(2) *DeepConf B10* averages the bottom-10% of group confidences across the trajectory, focusing on the least-confident segments. Letting $\mathcal{B}_{10}$ denote the index set of groups with the lowest 10% of $\{C_{G_i}\}$, which is equal to the lowest 10% per-step confidence set,

$$\text{DeepConf}_{\text{B10}}(\tau) \;=\; \frac{1}{|\mathcal{B}_{10}|} \sum_{i \in \mathcal{B}_{10}} C_{G_i}. \tag{10}$$

Both instantiations rely solely on the behavior policy probability $\tilde{\pi}^*$, sharing the same limitation as Self-Certainty in agentic settings: high local confidence from behavior policy alone may entangle generic linguistic priors with goal-directed progress.

LLM-as-a-Judge [75, 76, 77]. LLMs can be prompted as a judge to assign bounded scalar score assessments to outputs produced by other LLMs. This LLM-as-a-Judge paradigm has become a scalable substitute for human preference annotation in open-ended evaluation [78, 75, 79] and is increasingly used as a feedback source in model-development pipelines, including reward modeling and self-alignment [80, 81, 82]. More recently, judge models have also shown promise for evaluating goal-oriented, long-horizon agent trajectories, where the evaluator must reason over the full action-observation history rather than only the final response [83, 20]. In our UQ experiment[1], we use Claude-Sonnet-4.6 [52], a strong and relatively cost-effective modern LLM judge, to predict whether an agent succeeds in achieving the task goal using the prompt depicted in Figure 10.

WildReward [46]. We adopt `WildReward-8B` [46] as an off-the-shelf pre-trained outcome reward model[2], which is trained over massive user-chatbot interaction data, using it as a drop-in scorer across TTS and UQ for trajectory-level scoring. Given a trajectory input, it returns a single scalar reward estimate $\hat{r} := 1 + \sum_j \sigma(z_j) \in [1, 5]$ via CORAL ordinal regression [84] where z_j for $j = 1, 2, 3, 4$ denotes individual logit values from 4-way threshold classification head. Figure 11 depicts the prompt used for all the downstream evaluation.

ThinkPRM [47] is a PRM trained on mathematical reasoning data[3], that provides a step-by-step verification on the model-generation trajectory with per-step verdicts on correct-or-incorrect. We extract the per-step $P(\text{correct})$ from the vLLM inference logprobs and average across steps to derive trajectory-level reward estimate. We adapted the official prompt template tailored to the per-task setting: the template in Figure 12 for UQ and TTS and the template in Figure 13 for failure attribution.

AgentPRM [20]. As a task-specific training-based PRM baseline, we reproduce AgentPRM, which trains a reward head h_ϕ over the policy backbone to jointly predict each step's promise and progress under a combined regression objective. We follow the original recipe of the authors with `Qwen2.5-7B-Instruct` as both the behavior policy and the PRM backbone: we first SFT the behavior policy with LoRA ($r = 64$ and $\alpha = 128$) on 300 AgentTraj-L expert trajectories drawn

[1] Since the cost of API usage burdens a lot in the best-of-N sampling scenario, we confined these baselines to the UQ setup.
[2] https://huggingface.co/THU-KEG/WildReward-8B
[3] https://huggingface.co/launch/ThinkPRM-7B

from the AgentGym WebShop train split[4] (with the 100 test samples held out). Then, we continually LoRA fine-tune the backbone with a randomly initialized linear reward head, $h_\phi : \mathbb{R}^{3584} \to \mathbb{R}$, for 3 epochs at 1e-5 learning rate on 8000 on-policy trajectories sampled with temperature 1.0, with GAE parameter set $\lambda = 0.95$ and $\gamma = 1$.

B.2 Application-Specific Details

B.2.1 Test-time scaling (TTS)

Problem definition. We consider best-of-N (parallel) sampling [10, 85] as a testbed for reward models on a test-time compute scaling application, where we sample multiple responses (with non-zero generation temperature parameter) from an LLM given the same task prompt in parallel and score each trajectory to select the one that results in the *best* score among them for the actual evaluation. That is, given a trajectory τ starts from a prompt s and a task-specific (typically binary) success-failure evaluator $y(\tau \mid s) \in \{0, 1\}$, we measure average task success rate on a dataset $\mathcal{D}$ of task prompts as below,

$$\frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} y(\tilde{\tau} \mid s) \quad \text{for} \quad \tilde{\tau} = \arg\max_{\tau \in \mathcal{T}} r(s, \tau) \quad \text{with} \quad \mathcal{T} = \{M_\pi^{(i)}(\cdot|s, c)\}_{i=1}^{N}, \tag{11}$$

where $M_\pi^{(i)}(\cdot|s, c)$ denotes the i-th trajectory generated by an LLM M_π with a policy π given a temperature parameter $c \geq 0.0$ and $r(\cdot)$ denotes a reward score computed over the trajectory τ. The aim of this evaluation is to compare different reward models $r(\cdot)$ that yield the best success rate.

Benchmark. We adopt four different benchmarks that are specifically designed for evaluating LLMs' agentic capability: BFCLv4 [26], WebShop [27], AgentDojo [28], and τ^2-bench Airline [50, 29]. Below we elaborate on each benchmark, while deferring the description of the τ^2-bench to the next subsection B.2.2. Table 7 summarize the statistics

Table 7: **Test-time scaling benchmarks.** For each benchmark, we sample $N = 8$ rollouts per task at temperature c, then score every rollout with each reward scoring method and select the argmax. AgentDojo aggregates four suites; the per-suite task counts are listed in parentheses.

Benchmark	Split	# tasks	c
BFCLv4-MT [26]	Multi-turn base split	200	0.4
WebShop [27]	First 100 tasks subset with `items_shuffle_1000.json`	100	0.7
AgentDojo [28]	Workspace (40), Slack (21), Banking (16), and Travel (20)	97	0.4
τ^2-bench [50, 29]	Airline	50	0.7

BFCLv4 is a benchmark focusing on assessing an agent's function using capability. Here, the agent must emit a sequence of tool calls whose arguments and ordering exactly match a reference, with cross-turn argument propagation graded by a deterministic verifier. We use the `multi_turn_base` category (200 tasks) for the format-fidelity probing test while ignoring the harder long and missing-info categories.

AgentDojo is a benchmark of computer-use task suites with realistic mock APIs (reading email, posting to Slack, banking transactions, travel booking) where the agent must complete a free-form natural-language request by issuing tool calls. We run four suites, Workspace (40), Slack (21), Banking (16), Travel (20), totaling 97 tasks, where the success requires both the correct final state and a clean termination.

WebShop is a simulated e-commerce environment in which the agent navigates HTML-rendered product search and click pages to fulfill a shopping instruction conveyed through natural language. The reward is shaped in $[0, 1]$ based on attribute overlap with the target product. We evaluate on the first 100 tasks with 1000 products (`items_shuffle_1000.json` split) and 30 per-rollout interaction step limits, which is the standard WebShop split used in prior work.

Scenario-specific baseline. We consider three baselines in this best-of-N setup: greedy decoding, mean-of-N, and pass@N. Since the default inference mode of TTS is generation with non-zero

[4] For the experiment in Tab. 12, we used AgentGym wrapper while used WebShop codebase for other TTS experiments.

temperature, we consider a zero-temperature greedy decoding as a reference for each benchmark. This can be useful to check whether the non-zero temperature exploratory decoding is more helpful than a deterministic, exploitative decoding strategy on each task [86]. Meanwhile, mean-of-N is literally the average success rate of N candidate trajectories, representing the average competency of exploratory decoding. Finally, we also report pass@N as an oracle, upper bound score achievable by any selection method, indicating that the rate of at least one trajectory among N candidates succeeds.

B.2.2 Uncertainty quantification (UQ)

Problem definition. UQ is a representative application in the deployment-phase monitoring of LLM agents [87, 88, 51], which is becoming a central interest of research to realize trustworthy AI in the wild. Similar to the TTS problem setup, given the task prompt s, a trajectory-level reward score $r(s, \tau)$ predicts whether the trajectory τ will end in success as below, $\mathbb{I}\big(r(s, \tau) > H\big)$ given a classification threshold $H \geq 0$. By following the evaluation standard in LLM UQ research [89, 90], we report the area under the receiver operating characteristic curve (AUROC) which serves as a threshold-independent, balanced measure of binary prediction quality.

Benchmark. We adopt τ^2-bench [50, 29] Airline and Retail domains as our main testbed for UQ (Telecom domain was excluded from consideration since the modern agents' performances are saturated on that domain). Each domain has a detailed policy prompt that lists domain-specific constraints and ground rules, and the agent LLM equips that policy as a system prompt to define its domain-specific background context. In the meantime, the benchmark also hosts another LLM as a user simulator (we adopt Kimi-K2.5 [91] given its outstanding cost-effectiveness), who has a brief persona with synthetic personal information and also has a seed prompt to define their goal per task. Under this scaffolding, the agent spans a long-horizon trajectory by interacting with the user and pre-defined tools to achieve the complex goal[5]. Different to the TTS setup, we here fix the temperature parameters for both agent and user LLMs to zero to make the trajectory generation deterministic, i.e., greedy decoding mode, and quantify uncertainty once over these deterministic generation passes to prevent any unexpected confounding effects. The greedy decoding performance per-model in these two domains is provided in Table 8, where we found that the latest two model backbones `Gemma4-4B` and `Qwen3.5-9B` show balanced moderate performance on both domains.

Table 8: **τ^2-bench Airline and Retail greedy decoding success rate**. N denotes # of samples.

Domain	`Gemma4-4B`	`Qwen3.5-9B`	`Qwen3-14B`	`Olmo3-7B`
Airline ($N = 50$)	34.0	60.0	12.0	30.8
Retail ($N = 114$)	45.6	64.9	50.0	16.7

Scenario-specific baseline. We consider the LLM-as-a-Judge baseline, which prompts a powerful LLM to predict the binary outcome of success or failure given a realized agent trajectory. We adopt Claude-Sonnet-4.6 as our LLM judge, given its remarkable performance while requiring way cheaper API cost than its competitors. See the section B.1 for more description.

B.2.3 Failure attribution (FA)

Problem definition. We pose an agentic system that consists of multiple LLM agents, each equipped with tool-calling and inter-agent communication capabilities, collectively designed to solve a complex, long-horizon task. A trajectory $\tau \in \mathcal{T}$ generated from this agentic system is simply defined as $\tau = (s, a_1, a_2, ..., a_{T(\tau)})$ with the user's initial prompt s and sequence of agents' actions a_t with varying per-trajectory length $T(\tau)$. Given a failure trajectory τ and a ground truth error step annotator $y(\tau) = \{t \,:\, a_t$ is an incorrect action contributing to the system failure.$\}$, the failure attribution task aims to predict a *decisive error step*, $t_{\text{err}} := \min_t y(\tau)$ that becomes **the earliest critical error causing the failure of the system**. For all methods, we predict the decisive error step as the index of the steepest cummulative reward drop step $\hat{t}_{\text{err}} := \arg\min_t \sum_{i=0}^{t} r(s_i, a_i) - \sum_{i=0}^{t-1} r(s_{i-1}, a_{i-1})$ (which is equal to the minimum step reward index, $\arg\min_t r(s_t, a_t)$). Then, we measure the step-level prediction accuracy $\mathbb{I}(t_{\text{err}} = \hat{t}_{\text{err}})$ over the dataset.

[5] Some example trajectories can be explored in this official online leaderboard.

Benchmark. We adopt Who&When [30] benchmark where the seed tasks are drawn from two sources: (1) GAIA [92], which contains queries requiring information processing from multiple modalities, such as PDFs, spreadsheets, images, videos, and audio, as well as web browsing and coding capabilities; and (2) AssistantBench [93], which asks agents to play with multiple websites across various topics in geography, visual arts, biology, and so on. From the seed task prompt, trajectories were then generated by two agentic systems, CaptainAgent [94] and Magnetic-One [95], using GPT-4o as a behavior policy. The dataset consists entirely of failure trajectories, comprising 184 trajectories in total, each annotated with a decisive error step label.

Scenario-specific baseline. We consider AgenTracer [53] as a training-based method specifically trained on failure attribution tasks. We report the performance of Zhang et al. [53] which conducts GRPO training from `Qwen3-8B` base model on 2.5K failure trajectories with step-level annotations. Given an arbitrary-length trajectory, AgenTracer predicts a decisive error step with a reasoning trace.

B.3 Inference Setup and Configuration

The aforementioned three classes of experiments, TTS, UQ, and FA, share the following common configuration unless noted. All LLMs are loaded in `bfloat16` on a single GPU with model-family-tailored tokenizers and chat templates (e.g. `qwen3_coder` & `qwen3` reasoning parser for Qwen3.5; `gemma4` for Gemma4; `olmo3` for Olmo3).

In **TTS** setup, we use vLLM [96] to generate trajectory with `max_model_len`=32768 and set `enable_thinking=false` for all model backbones (except the Olmo3) that support inference mode selection. Greedy decoding baselines adopt zero generation temperature and `top_p`=1.0; the best-of-N stochastic trajectory generation set `top_p`=0.95, `max_new_tokens`=1024, and temperature as 0.7 on τ^2-bench and WebShop, whereas BFCL and AgentDojo adopt 0.4 temperature. The user simulator in τ^2-bench is held at zero-temperature, so the inter-trial volatility comes from the agent only.

In **UQ** and **FA**, we score pre-generated greedy decoding trajectories by re-tokenizing the full conversation log and running a single forward pass per trajectory through the standard HuggingFace transformers library as the engine. We left-truncate to a context window of `max_length`=16384 tokens and read the per-position log-softmax for confidence-based and our method.

C Additional Results

Is top-k averaging over log token probability helpful? We first compare our default progress advantage, $\log \tilde{\pi}^*(a|s) - \log \pi_{\text{ref}}(a|s)$, with a top-$k$ smoothed version, progress k-advantage. Table 9 and Table 10 present the results in TTS and UQ, respectively, and Figure 6 provides results on the FA.

Table 9: **TTS through best-of-8 sampling**. We compare reward scoring methods across two LLM backbones, reporting the success rate (%) of the selected trajectory averaged across four datasets (BFCLv4, WebShop, AgentDojo, and τ^2-Airline).

Scoring Method	`Gemma4-4B` Avg.	`Qwen3.5-9B` Avg.
Pass@N (oracle)	45.4	67.5
Greedy Decoding	33.4	54.6
Mean-of-N	33.1	54.7
WildReward-8B [46]	33.1	54.8
ThinkPRM-7B [47]	30.9	52.2
ThinkPRM-14B [47]	33.6	54.9
Self-Certainty [48]	29.0	51.5
DeepConf Tail [49]	31.4	53.2
DeepConf B10 [49]	27.4	55.8
Progress k-Advantage	34.1	58.1
Progress Advantage	**38.8**	**62.1**

Table 10: **UQ for trajectory monitoring**. We compare scoring methods across four LLM backbones on τ^2-bench Airline and Retail to predict an agent's success on each model's greedy-decoding trajectory. AUROC averaged across the four backbones (`Gemma4-4B`, `Qwen3.5-9B`, `Qwen3-14B`, and `Olmo3-7B`).

Scoring Method	τ^2-Airline Avg.	τ^2-Retail Avg.
✳ Sonnet-4.6 [52]	0.644	0.818
WildReward-8B [46]	0.420	0.596
ThinkPRM-7B [47]	0.457	0.558
ThinkPRM-14B [47]	0.520	0.591
Self-Certainty [48]	0.658	0.441
DeepConf Tail [49]	0.581	0.429
DeepConf B10 [49]	0.669	0.446
Progress k-Advantage	0.732	0.646
Progress Advantage	**0.781**	**0.671**

We observe that both the progress advantage and k-smoothing variant greatly outperform the baseline methods on both TTS and UQ scenarios by consistently achieving the best and the second-best performances. In these two scenarios, the default progress advantage beats the k-smoothing version,

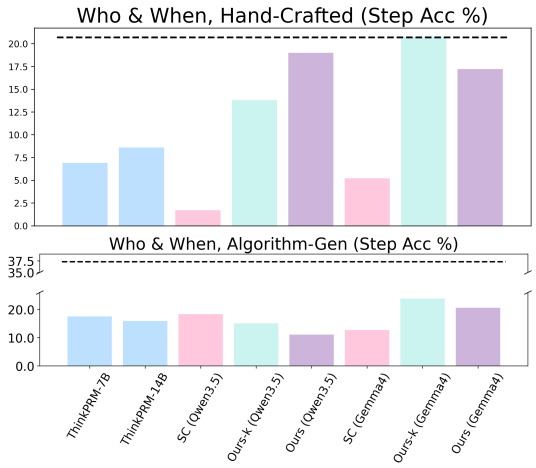

Figure 6: **Who & When step-level accuracy**. We predict when the agent system makes a decisive error. SC denotes Self-Certainty [48], Ours-k denotes the progress k-advantage, and the dashed line denotes AgenTracer [53] specifically trained on this failure attribution task through RL-training.

showing its effectiveness grounded in theoretical derivation. Meanwhile, progress k-advantage exceeds the default progress advantage on some of the FA setups with `Gemma4-4B` backbone, wherein the precise per-step credit assignment is crucial, suggesting the smoothed representation of probability may sometimes be better than the exact one to model step-level progress. This result implies that one can carefully choose the representation of token probability (pure token probability, k-averaged, etc.) for the intended use and characteristics in a downstream task and data.

Comparison between progress advantage and its ingredients. Since our progress advantage is constructed with the log probability of the behavior policy $\log \tilde{\pi}^*(a|s)$ and the reference policy $\log \pi_{\text{ref}}(a|s)$, the natural question is whether we should blend the two log probabilities rather than using one of them for simplicity and efficiency. In Figure 3, we showed how the pure policy log probability $\log \tilde{\pi}^*(a|s)$ results in poor scoring, while progress advantage derives desirable scoring through a qualitative analysis. In this paragraph, we further provide a summary of quantitative results on the comparison between the progress advantage $\log \frac{\tilde{\pi}^*(a|s)}{\pi_{\text{ref}}(a|s)}$, behavior policy log probability $\log \tilde{\pi}^*(a|s)$, and reference policy log probability $\log \pi_{\text{ref}}(a|s)$ across eight UQ scenarios. Table 11 shows the average ranking and corresponding average AUROC. We see that the progress advantage outperforms its two ingredients by large margins.

Table 11: **Comparison between the progress advantage and its ingredients.** On the eight UQ scenarios (two domains and four model backbones), we compute the average ranking and AUROC of each reward signal across all possible combinations of token and step aggregations (25 in total) per method. We see that progress advantage consistently outperforms the individual ingredients, such as the log probability of the behavior policy and reference policy.

Scoring Method	Avg. Rank by Best AUROC	Avg. Best AUROC
$\log \frac{\pi_\theta(a\|s)}{\pi_{\text{ref}}(a\|s)}$	**1.44** $\pm$ 0.62	**0.732** $\pm$ 0.059
$\log \pi_\theta(a\|s)$	2.25 $\pm$ 0.89	0.679 $\pm$ 0.044
$\log \pi_{\text{ref}}(a\|s)$	2.31 $\pm$ 0.70	0.695 $\pm$ 0.043

Extended results on advantage aggregation strategies. As mentioned in Section 3.3 and Section 4.5, aggregation of the advantages across tokens and steps significantly affects the downstream utility of progress advantage. For the best-of-N scenarios (Figure 7), the best working aggregation strategy is not universal across datasets or models. Some datasets, such as WebShop and AgentDojo, show somewhat robust results across different aggregation methods. For example, choosing MAX operator as a token aggregation and LAST or MIN as step aggregation yields promising results for `Gemma4-4B` on WebShop, whereas MEAN or MIN as token aggregation and LAST as step aggregation

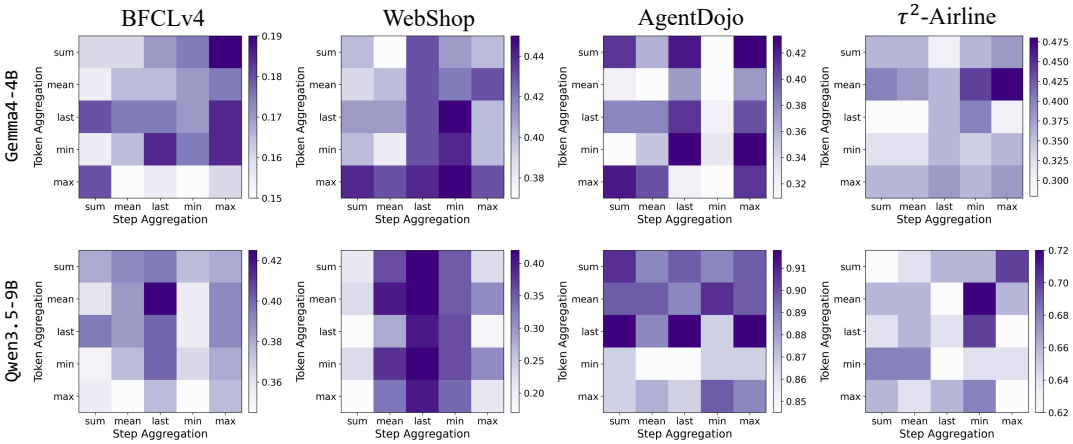

Figure 7: **Varying combinations of token and step aggregation strategy for progress advantage in best-of-N.** We sweep 25 combinations of token-wise and step-wise aggregation of progress advantage over four datasets and two model backbones in the best-of-8 scenario.

become the winning tickets for `Qwen3.5-9B`; other datasets exhibit sensitivity depending on the aggregation where (MEAN, LAST) as the (token, step) aggregation pair stands for the sole winner in BFCLv4 dataset with `Qwen3.5-9B`.

In the meantime, the trend in UQ (Figure 8) is better interpretable, where we see (MAX, MEAN) combination is a winning strategy for Airline, whereas (MIN, LAST) combination is a winner for Retail. We can conclude that extreme tokens that produce maximum or minimum progress advantage are informative to define a per-step signal in UQ. Meanwhile, the effectiveness of step-wise aggregation reflects the domain structures: most of **tasks in the retail domain** (*return*, *exchanges*, *order modification*) have some clear terminating action, and a single tool call can commit the outcome sometimes—incentives LAST as a suitable operation. Meanwhile, most of **tasks in the airline domain** (*cancellations*, *baggage policy*, *membership status checks*) are kind of policy negotiations where the agent and user talk through the decision over multiple steps, and the success/failure is a slow event distributed across the whole conversation, making MEAN as a go-to operation.

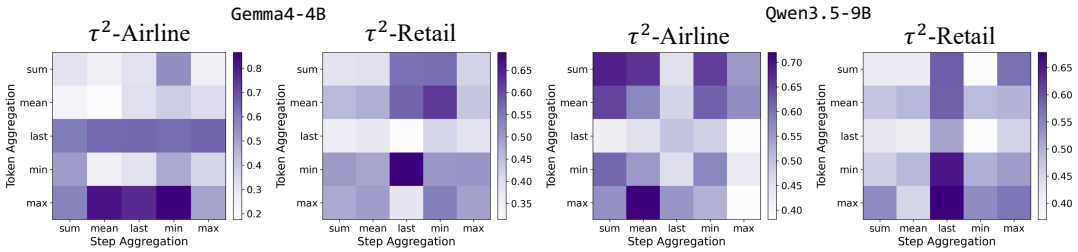

Figure 8: **Varying combinations of token and step aggregation strategy for progress advantage in UQ.** We sweep 25 combinations of token-wise and step-wise aggregation of progress advantage over two domains in τ^2-bench and two model backbones in the uncertainty quantification scenario.

Visualization of progress advantage evolution. We have observed promising results of progress advantage in the UQ setup so far. To dive deeper into this success, we analyze how the progress advantage actually evolves step-by-step across the whole trajectory. In Figure 9, we visualize group average per-step progress advantage for success and failure trajectory groups, where we normalize the step index to the [0,1] range since each trajectory has a different length. In the Airline domain, we see that progress advantage clearly separates the two groups from the very beginning. In the Retail domain, on the other hand, it shows a tied trend most of the time, but gives a higher advantage to the success group at the very end, implying that while the progress advantage shows its effectiveness broadly, the way it works can vary across domains.

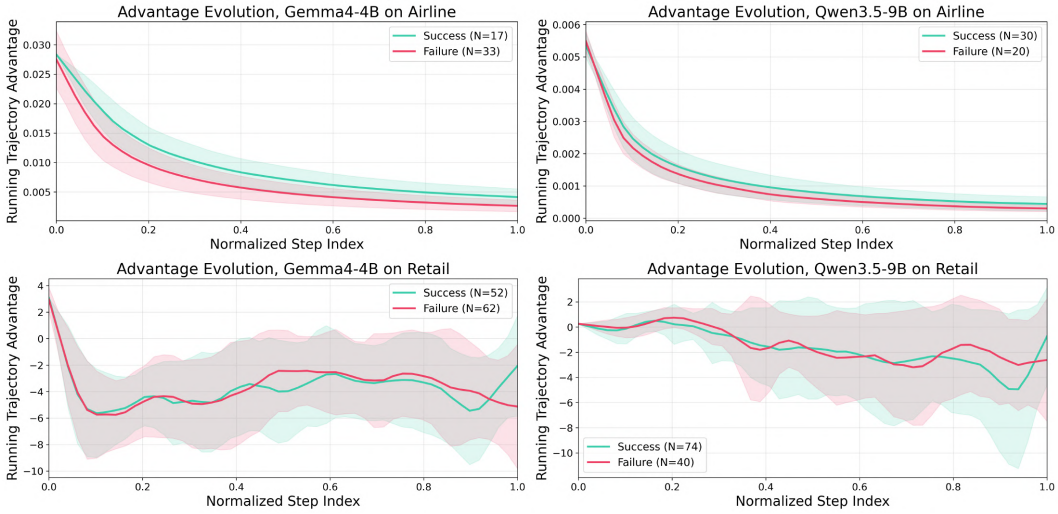

Figure 9: **Progress advantage evolution across trajectory**. We visualize group average per-step progress advantage over the τ^2-bench greedy decoding trajectories generated by `Gemma4-4B` and `Qwen3.5-9B` where we apply MAX and MIN aggregation across tokens within each step for Airline and Retail domains, respectively, and apply MEAN and LAST aggregation across steps for Airline and Retail domains to get the running advantage. The group is defined by binary success and failure outcomes. The shade denotes one standard deviation across within-group per-step progress advantage.

Table 12: **Test-time scaling through best-of-8 sampling on WebShop.** Given eight sample trajectories from `Qwen2.5-7B-Instruct` as a behavior policy with temperature 0.7, we compare training-based reward models as well as training-free confidence-based methods to the default progress advantage and its GRPO task-specific fine-tuned variant.

Method	Training	Success Rate
PassN (oracle)	✗	45.0
Greedy Decoding	✗	30.0
Mean-of-N	✗	31.0
WildReward-8B [46]	✓	32.0
ThinkPRM-7B [47]	✓	32.0
ThinkPRM-14B [47]	✓	<u>35.0</u>
AgentPRM-7B [20]	✓	33.0
Self-Certainty [48]	✗	30.0
DeepConf Tail [49]	✗	29.0
DeepConf B10 [49]	✗	27.0
Progress Advantage	✗	<u>35.0</u>
Progress Advantage w/ GRPO	✓	**38.0**

Comparison with AgentPRM. Due to the resource constraint, we mainly considered WildReward [46] and ThinkPRM [47] in our main paper as training-based PRM baselines, which are trained on general multi-turn interaction or reasoning datasets rather than the actual downstream task datasets. In this paragraph, we provide results with an additional PRM baseline, AgentPRM [20], **which is directly trained on a specific downstream task**, e.g., WebShop, with a value head to predict the Q-value per step. Specifically, we follow the recipe in AgentPRM by first performing a small-scale SFT from `Qwen2.5-7B-Instruct` and then going through the actual reward modeling with step-wise advantage estimations.

In Table 12, although AgentPRM beats the comparable-size general pre-trained reward models, WildReward-8B and ThinkPRM-7B, as well as confidence-based baselines, it largely underperforms

our progress advantage that was not trained on this specific WebShop dataset. This implies a substantial challenge to building reliable PRMs on agentic tasks, where we offer a new angle on this.

Can progress advantage take benefit of task-specific RL training? In the main body of the paper, our core statement was that the progress advantage automatically emerges after general post-training, and it is a sufficiently useful signal to guide or monitor agentic inference. Meanwhile, one may wonder if we could further make the signal sharper and tailored to a specific downstream task by further RL fine-tuning the behavior policy on a task of interest. In Table 12, we present the results on this by training `Qwen2.5-7B-Instruct` with GRPO [8] on the WebShop dataset under the hyperparameter specification in Table 13. We see that progress advantage without any task-specific training already achieves the best performance, rivaling the ThinkPRM-14B, which doubles model size compared to the behavior or reference policy; Besides, constructing the progress advantage with $\log \pi_{\text{GRPO}}(a|s) - \log \pi_{\texttt{Qwen2.5-7B-Instruct}}(a|s)$ further pushes the success rate about 8.5% by sharpening the reward signal to downstream tasks.

Table 13: **Hyperparameter configuration for GRPO training on WebShop**.

Hyperparameter	Value
Max prompt length	4096
Max response length	512
Max environment steps	10
Learning rate	2×10^{-6}
Success reward	10
Failure reward	0
Invalid action penalty	-0.1
Group size	8
Rollout temperature	1.0
Validation temperature	0.4
Mini-batch size	64
KL coefficient	0.01

D Derivation of Implicit Rewards Under Stochastic MDP

The KL-constrained reward maximization problem is formulated as follows,

$$\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} \Big[\sum_t r(s_t, a_t) - \beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \Big| s_0 \sim \rho \Big], \tag{12}$$

where $\beta > 0$ is the regularization coefficient and π_{ref} denotes a reference policy, commonly built with pre-trained or SFT checkpoints. This objective can be equally expressed as the following maximum entropy RL form [97, 98] with entropy $H(\cdot)$,

$$\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} \Big[\sum_t r(s_t, a_t) + \beta \log \pi_{\text{ref}}(a_t|s_t) + \beta H\big(\pi_\theta(\cdot|s_t)\big) \Big| s_0 \sim \rho \Big]. \tag{13}$$

This optimization problem gives us a known solution [36, 24], $\pi^*(a_t|s_t) = \exp\big(\frac{Q^*(s_t,a_t) - V^*(s_t)}{\beta}\big)$, corresponding optimal value function $V^*(s_t) = \beta \log \sum_a \exp\big(Q^*(s_t, a)/\beta\big)$, and also the corresponding optimal action-value function given the Bellman optimality equation,

$$Q^*(s_t, a_t) = \begin{cases} r(s_t, a_t) + \beta \log \pi_{\text{ref}}(a_t|s_t) + \mathbb{E}_{s_{t+1} \sim f}[V^*(s_{t+1})], & \text{if } s_{t+1} \text{ is not terminal} \\ r(s_t, a_t) + \beta \log \pi_{\text{ref}}(a_t|s_t), & \text{otherwise.} \end{cases} \tag{14}$$

Now we re-express the Eq 14 as a reward-centric form and sum it across the trajectory up to $T-1$ position as below,

$$\begin{aligned}\sum_{t=0}^{T-1} r(s_t, a_t) &= \sum_{t=0}^{T-1} \big(Q^*(s_t, a_t) - \beta \log \pi_{\text{ref}}(a_t|s_t) - \mathbb{E}_{s_{t+1}}[V^*(s_{t+1})]\big) \\ &= \sum_{t=0}^{T-1} \big(\big[\beta \log \pi^*(a_t|s_t) + V^*(s_t)\big] - \beta \log \pi_{\text{ref}}(a_t|s_t) - \mathbb{E}_{s_{t+1}}[V^*(s_{t+1})]\big) \\ &= \sum_{t=0}^{T-1} \big(\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + V^*(s_t) - \mathbb{E}_{s_{t+1}}[V^*(s_{t+1})]\big). \end{aligned} \tag{15}$$

Eq. 15 indicates that we can no longer cancel out the intermediate value terms through the telescoping sum under stochastic MDP, and thus can not represent the exact reward solely with the known policy terms. This motivates us to explore an alternative derivation, progress advantage in Proposition 1.

E Missing Proof

E.1 Derivation of Progress Advantage

In this section, we provide a full derivation from the optimal policy, the optimal state value function, the optimal action value function, and finally, the optimal advantage as an implicit process reward.

Proposition 3 (Restatement of Proposition 1)**.** *Let $\tilde{\pi}^*$ be an optimal policy under the KL-regularized RL objective (Eq. 1) with $\beta > 0$, shaped with the reference policy π_{ref} where $\pi_{\text{ref}}(a|s) > 0$ for any $a \in \mathcal{A}$ and $s \in \mathcal{S}$. Then, the optimal advantage function is exactly recovered by the log probability ratio between $\tilde{\pi}^*$ and π_{ref} for any state and action:*

$$\tilde{A}^*(s, a) = \tilde{Q}^*(s, a) - \tilde{V}^*(s) = \beta \log \frac{\tilde{\pi}^*(a|s)}{\pi_{\text{ref}}(a|s)}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{16}$$

Proof. We start from the KL-regularized RL objective given π_{ref} and ρ without a discounting factor,

$$\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta, s_t \sim f}\big[\sum_{t=0}^{\infty} r(s_t, a_t) - \beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \big| s_0 \sim \rho\big], \tag{17}$$

where we construct the infinite sum of any finite T-length sequence by having an absorbing state with reward zero. This reward maximization problem can be also expressed as expected value maximization, $\max_{\pi_\theta} J(\pi_\theta) = \max_{\pi_\theta} \mathbb{E}_{s_0 \sim \rho}[\tilde{V}^{\pi_\theta}(s_0)]$, by definition of the state value function,

$$\tilde{V}^{\pi_\theta}(s) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)}\big[\sum_{t=0}^{\infty} r(s_t, a_t) - \beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \big| s_0 = s\big]. \tag{18}$$

Given the time-homogeneous transition probability $f(s'|s, a)$, by unrolling one step of the trajectory, we have

$$\tilde{V}^{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi_\theta(\cdot|s)}\big[r(s, a) - \beta \log \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)} + \mathbb{E}_{s' \sim f(\cdot|s,a)}[\tilde{V}^{\pi_\theta}(s')]\big]. \tag{19}$$

by renaming a_0 as a and s_1 as s'. It satisfies its own (soft) Bellman equation augmented by the KL penalty. Now, we have the following action value function by definition:

$$\tilde{Q}^{\pi_\theta}(s, a) = r(s, a) + \mathbb{E}_{s' \sim f(\cdot|s,a)}[\tilde{V}^{\pi_\theta}(s')]. \tag{20}$$

Then, plugging Eq. 20 into Eq. 19, we get the following for any π_θ and s.

$$\tilde{V}^{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi_\theta(\cdot|s)}\big[\tilde{Q}^{\pi_\theta}(s, a) - \beta \log \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)}\big]. \tag{21}$$

Since the above equation holds for any π_θ and s, we can now re-express our objective at state s given a fixed optimal action value function $Q^*(s, a)$ as follow,

$$\max_{\pi_\theta} \tilde{J}(\pi_\theta) := \max_{\pi_\theta} \sum_a \pi_\theta(a|s)\big[\tilde{Q}^*(s, a) - \beta \log \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)}\big], \quad \text{s.t.} \quad \sum_a \pi_\theta(a|s) = 1. \tag{22}$$

The optimal policy for this local step-level objective is equivalent to that of the global trajectory-level objective by the Policy Improvement Theorem [99, 41]. We then solve this constrained optimization problem with the method of Lagrangian multipliers to get the optimal policy,

$$\tilde{J}(\pi_\theta, \lambda) = \sum_a \pi_\theta(a|s)\big[\tilde{Q}^*(s,a) - \beta \log \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)}\big] + \lambda\big(1 - \sum_a \pi_\theta(a|s)\big) \tag{23}$$

$$\frac{\delta \tilde{J}(\pi_\theta, \lambda)}{\delta \pi_\theta(a|s)} = \tilde{Q}^*(s,a) - \beta(\log \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)} + 1) - \lambda \equiv 0 \tag{24}$$

$$\tilde{\pi}^*(a|s) = \pi_{\text{ref}}(a|s) \exp(\frac{1}{\beta}\tilde{Q}^*(s,a)) \exp(-\frac{\lambda}{\beta} - 1) \tag{25}$$

$$= \frac{1}{Z(s)} \pi_{\text{ref}}(a|s) \exp(\frac{1}{\beta}\tilde{Q}^*(s,a)), \tag{26}$$

where $Z(s) = \sum_a \pi_{\text{ref}}(a|s) \exp(\frac{1}{\beta}\tilde{Q}^*(s,a))$. Applying log-linearization induces the following,

$$\beta \log \frac{\tilde{\pi}^*(a|s)}{\pi_{\text{ref}}(a|s)} = \tilde{Q}^*(s,a) - \beta \log Z(s) \tag{27}$$

Plugging this Eq. 27 into Eq. 21 induces the following optimal state value function,

$$\tilde{V}^*(s) = \sum_a \tilde{\pi}^*(a|s)[\beta \log Z(s)] = \beta \log Z(s) \sum_a \tilde{\pi}^*(a|s) = \beta \log Z(s), \tag{28}$$

where $\tilde{Q}^*(s,a)$ was canceled. With Eq. 27 and Eq. 28, we finally get our optimal advantage function,

$$\beta \log \frac{\tilde{\pi}^*(a|s)}{\pi_{\text{ref}}(a|s)} = \tilde{Q}^*(s,a) - \tilde{V}^*(s) = \tilde{A}^*(s,a), \tag{29}$$

defined solely by the log-probability ratio. The stochasticity of the state transition $s' \sim f(\cdot|s,a)$ is embedded in Q^* and V^* under this general non-deterministic MDP. If we want to model one of them separately or want to directly model the reward function, we have to explicitly deal with that stochasticity. By embracing the optimal advantage function as a pseudo reward, we bypass the explicit stochasticity modeling while leaving its implicit reflection to the log-probability term computed solely from the realized observation. □

E.2 Proof: Clipping Surrogate RL as an Implicit KL Constraint

Proposition 4 (Restatement of Proposition 2)**.** *Let π_{ref} and π_θ be the reference and target policies sharing the same support. Define the importance sampling ratio as $R(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)}$. If optimization enforces a per-sample constraint $R(s,a) \in [1-\varepsilon, 1+\varepsilon]$ for all (s,a) and a small $\varepsilon > 0$, then $D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) \leq \frac{\varepsilon^2}{2}$ and $D_{\text{KL}}(\pi_{\text{ref}} \parallel \pi_\theta) \leq \frac{\varepsilon^2}{2}$, similarly for reverse KL, locally at $R(s,a) \approx 1$.*

Proof. Let $\delta(s,a) = R(s,a) - 1$. Then, the clipping surrogate RL optimization problem enforces $|\delta(s,a)| \leq \varepsilon$. We will show that any policy found under PPO-Clip style surrogate optimization is strictly within a KL (both reverse and forward) trust region of radius $\epsilon^2/2$.

First, the reverse KL divergence is defined as $D_{\text{KL}}(\pi_{\text{ref}} \parallel \pi_\theta) = \mathbb{E}_{\pi_{\text{ref}}}[-\log R]$. The second-order Taylor expansion of $-\log R = -\log(1+\delta)$ around $R = 1$ yields:

$$-\log(1+\delta) = -\delta + \frac{\delta^2}{2} + \mathcal{O}(\delta^3) \tag{30}$$

Similarly, the forward KL divergence is defined as $D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) = \mathbb{E}_{\pi_\theta}[\log R]$. With an importance sampling ratio R, it can be equivalently expressed as $\mathbb{E}_{\pi_{\text{ref}}}[R \log R]$. The second-order Taylor expansion of $R \log R = (1+\delta)\log(1+\delta)$ around $R = 1$ yields:

$$(1+\delta)\log(1+\delta) \approx (1+\delta)\left(\delta - \frac{\delta^2}{2}\right) = \delta + \frac{\delta^2}{2} + \mathcal{O}(\delta^3) \tag{31}$$

Now, taking the expectation over π_{ref} for both expansions, Eq. 30 and Eq. 31, we have:

$$D_{\text{KL}}(\pi_{\text{ref}} \parallel \pi_\theta) \approx \mathbb{E}_{\pi_{\text{ref}}}\left[-\delta + \frac{\delta^2}{2}\right], \quad D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) \approx \mathbb{E}_{\pi_{\text{ref}}}\left[\delta + \frac{\delta^2}{2}\right] \tag{32}$$

Since π_{ref} and π_θ share the support and are valid probability distributions, $\mathbb{E}_{\pi_{\text{ref}}}[R] = \mathbb{E}_{\pi_{\text{ref}}}[\frac{\pi_\theta}{\pi_{\text{ref}}}] = 1$. That means, $\mathbb{E}_{\pi_{\text{ref}}}[\delta] = 0$, i.e., the exact removal of the linear term, collapsing both KL divergences to the scaled Pearson χ^2-divergence:

$$D_{\text{KL}} \approx \frac{1}{2}\mathbb{E}_{\pi_{\text{ref}}}[\delta^2] \tag{33}$$

Given the per-sample constraint $|\delta(s, a)| \leq \varepsilon$, we have $\mathbb{E}_{\pi_{\text{ref}}}[\delta^2] \leq \varepsilon^2$. Then, substituting this into our approximation, Eq. 33, yields below:

$$D_{\text{KL}} \lesssim \frac{\varepsilon^2}{2} \tag{34}$$

for both the forward and reverse directions. Therefore, any feasible policy under the PPO-Clip style constrained RL becomes a proper subset of a KL trust region of radius $\delta = \frac{\varepsilon^2}{2}$, regardless of whether the objective applies a forward or reverse KL penalty. □

F Limitation and Future Work

Since institutions usually do not release all intermediate checkpoints from their LLM development pipelines, we inevitably limited the scope of model candidates in our experiments. We hope that the community pursues the fully open model development cycle in the future for novel uses like our progress advantage. Also, although we assumed that the public, post-trained models are close to the optimal solution of the RL objective, it is barely falsifiable since we usually don't know the full training configuration details, as well as the training log. Future work with controlled experiments to validate the approximation quality of progress advantage would be appealing. Besides, although we established a theoretical foundation on the implicit process reward for agents in stochastic MDP, we leave vague the engineering efforts to find the best working specifications and implementations of progress advantage for future work.

G Broader Context and Discussion

G.1 Outcome Reward Modeling

The early driving force in preference learning and reinforcement learning of LLMs was the outcome reward models (ORMs) [5, 7, 6]. They provide supervision at the level of final responses, rewarding reasoning trajectories according to the quality of their ultimate outcomes, regardless of their process. The annotations over outcomes are relatively easy to collect, making ORMs a scalable choice for reranking, search, and RL for reasoning models [100, 8]. Nevertheless, ORM supervision is coarse and delayed [10, 101]: it evaluates only the final outcome and does not verify whether the intermediate reasoning process is sound. Consequently, trajectories that contain errors or spurious steps may still be rewarded if they happen to produce the correct answer. These limitations motivate process reward models (PRMs), which instead provide fine-grained supervision over intermediate reasoning steps. We propose a practical approach to building PRMs on agentic scaffolding to utilize the fine-grained signals in broad inference-time applications.

G.2 Implicit Reward Modeling

After seminar works published [23, 102, 24], implicit reward formulation has gained its bold popularity due to its practical merit, allowing users to bypass explicit reward modeling. The implicit reward-based approach has usually been developed in a preference fine-tuning setup [40, 39, 37, 38, 103] but also extended to general multi-step reasoning or multi-step interaction setups recently [104, 105, 21, 68, 22]. However, its remain opaque to adopt this implicit reward modeling technique on the LLM agent setup where the rollouts of the agents are made up with not only the model's own deterministic token completion but also with stochastic observations from the environmental entities. To fill this gap, we establish a foundation of implicit reward under stochastic MDP tied with the realistic LLM agents inference settings, by deriving the progress advantage formulation.

G.3 Distribution Contrasting and Sharpening

In essence, progress advantage contrasts the likelihood of a behavior policy with that of a reference policy. This kind of contrastive probabilistic quantity has a rich history in statistics and the machine learning field. For instance, in its vanilla form, one can easily draw an interpretation as a likelihood ratio test statistic [106, 107] as well as a logistic regression model in the noise contrastive estimation [108, 109, 110]. If we take the expectation over it, given a distribution, it becomes a measure of contrastive divergence [111, 112, 113] with the KL divergence instantiation. In addition to this luxuriant connection with the classic, progress advantage has also dense relationships with contemporary findings in LLM research, e.g., *contrastive decoding* [54, 114] that builds a keen output token probability distribution by contrasting the probabilities of an expert and an amateur model, as well as an interpretation of RL post-training as a *distribution sharpening* [115, 116]. Despite this wide spectrum of relevant literature, it still remains untapped to use this contrastive likelihood quantity as an implicit reward signal under agentic harness. We expect our progress advantage and its connection to the broader concepts we discussed here to present insights and exciting follow-ups.

G.4 Open-source AI and Sustainable Machine Learning

By envisioning the promise of updatable and sustainable machine learning [117], some researchers have explored how artifacts produced during the model development process, e.g., model weight checkpoints, can be shared across community members and reused efficiently [118, 119, 120]. This line of work has renewed interest in the classical model ensemble [121, 122, 123, 124, 125] and has recently led to substantial progress under the term *model merging* [126, 60, 63, 127]. Another line of work is *black-box prompt optimization* [128, 129, 130, 131], which solely leverages the output probabilities from the black-box models to customize the model's input space interface without knowledge about model architectures or parameters. By making the development process and artifacts more transparent and broadly accessible, we can reduce the waste of repeatedly "reinventing the wheel", while creatively leveraging the existing common goods to advance our own intelligence systems. We hope that our progress advantage inspires such creative adoption of existing resources. Note that the pursuit of this kind of sustainable machine learning is not limited to the reuse of model weights or outputs. We may need to revisit the entire development pipeline [132, 72], i.e., from the earliest stage of data collection, to find out whether **neglected free lunches** remain [133].

H Broader Impacts

Progress advantage offers a way to score trajectories from agentic systems without undergoing the cost-heavy development phase. This can contribute to reducing the total GPU hours from groups that may want to adopt progress advantage during their LLM deployment under agentic harness, while assisting safety and outlier monitoring, as well as improving the system response quality. Meanwhile, since the progress advantage produces reward signals based on the log probability ratio of trained LLM policies, it may contain historical and social bias hidden in the pre/mid/post-train datasets, resulting in biased scoring and preference during its deployment-time applications. Therefore, one should take care of the trajectories gone through the progress advantage monitoring before the public release by leveraging some well-established safeguards tools [134, 135, 136].

I Computing Resource Statement

We used NVIDIA A6000, A100-SXM4, and H200 in a mix, but reported the GPU hours in terms of a single A100. To reproduce our progress advantage (as well as component-wise ablation) in best-of-N experiments of the TTS scenario, the full sweep may require 26 hours; UQ may require 16 hours, and the FA will not take much, up to 4 hours, resulting in 46 hours in total for only our method. Entire replication, including ours and the full baseline, and the k-smoothed variant of progress advantage, would take four times longer. Besides, the rough per-model (and per-pair) memory requirement dominated by the weight footprint for progress advantage is described in Table 14.

Table 14: **bf16 weight footprint per backbone pair (policy and reference).**

Backbone	Per-model	Pair peak
Qwen3.5-9B	∼18 GB	∼36 GB
Qwen3-14B	∼28 GB	∼56 GB
Qwen2.5-7B	∼14 GB	∼28 GB
Gemma4-4B	∼8 GB	∼16 GB
Olmo3-7B	∼14 GB	∼28 GB

J Prompt Template for Baseline Methods

```
You are a binary success predictor for LLM agent trajectories. Given the
    agent-user interaction trajectory, predict whether the agent succeeds.

Trajectory: \n {trajectory}

Output success = 1 if the agent completes the user's goal correctly.
Output success = 0 if the agent fails, leaves the task incomplete,
violates policy, makes a material factual/tool-use error, or causes
a wrong outcome.

When uncertain, choose the more likely label and report confidence.

Return JSON only:

{
  "success": <0 or 1>,
  "confidence": <number between 0.0 and 1.0>,
  "justification": "<one or two sentences explaining the prediction>"
}
```

Figure 10: Prompt used for the LLM-as-a-Judge (Claude-Sonnet-4.6 [52]) baseline in UQ experiment.

```
# Task Description
You are an expert conversation evaluator. Your task is to judge the
**User's Satisfaction** with the Assistant's response based on the
conversation context.
Please rate the response on a scale of 1 to 5 integers.

# Scoring Criteria
[1] CLEARLY NEGATIVE / REJECTION
[2] CORRECTION / ERROR POINTER (Negative)
[3] NEUTRAL
[4] POSITIVE ENGAGEMENT
[5] CLEAR SATISFACTION

# Input Data
## Context (History)
{history}

## User Query
{query}

## Assistant Response
{response}

# Output
Based on the criteria above, please output ONLY the integer score
(1, 2, 3, 4, or 5).
```

Figure 11: Prompt template used for the outcome reward model baseline `WildReward-8B` across all downstream applications. Note that the trajectory-level reward is defined as $1 + \sum_{j=1}^{4} \sigma(z_j)$, the sum of four ordinal regression threshold logits z_j obtained directly from the model's classification head; we did not actually generate the integer that the prompt requests.

```
You are given a task and a proposed step-by-step solution:

[Task] \n {task}

[Solution] \n {solution}

Review and critique each step in the proposed solution to determine
whether each step is correct. If the solution is incomplete, only
verify the provided steps. For each step, end your critique with a
single line of the form 'Step N is {correct}' or
'Step N is {incorrect}'.
```

Figure 12: Prompt template used for `ThinkPRM` in TTS and UQ settings. `{task}` is the initial user query; `{solution}` is the agent's task-solving trajectory interacting with tools and/or the user.

```
You are given a multi-agent problem-solving trajectory
and must verify each step.

[Task] \n {task}

[Solution] \n {solution}

Review and critique each step in the proposed solution to determine
whether each step is correct. For each step, end your critique with
a single line of the form 'Step N is {correct}' or
'Step N is {incorrect}'.
```

Figure 13: Prompt template used for `ThinkPRM` in the FA setting.