CIS 6200: Emerging Theory of LLMs **Fall 2026**

Topic 5D: Process Rewards and Guided Generation

Last updated: October 26, 2026 *Surbhi Goel*

Acknowledgements: *The analysis of guided sampling and backtracking follows Rohatgi et al. [1]. The training connection draws on VinePPO [3]. The discussion of process supervision draws on Lightman et al. [4] and Cheng et al. [5].*

AI Disclosure: *These notes were drafted and revised with AI assistance. If you find an error, please let me know.*

Best-of-N sampling scores and selects among complete responses. Could feedback on an unfinished response help us spend less computation on continuations that will fail? During training, there is a related problem: REINFORCE from the previous lecture on policy gradients gives every step the same final reward, even when a late mistake spoils a useful beginning.

A **process verifier** evaluates intermediate steps in context. We study a particular kind of process score: the probability that the model will eventually succeed from the current prefix. First we use these continuation values to guide generation. Exact values give a simple sampler. Errors in the values motivate allowing it to revise earlier choices. We then use continuation values to assign credit during training. We discuss how to obtain these values at the end.

1 Process scores and continuation values

Fix a prompt x and suppress it in the notation. Let q_{ref} be a fixed model with positive next-step probabilities (the generator previously denoted q_0).

1.1 The target distribution

We want successful responses while retaining the variety of the base model. For example, generating the same valid test input repeatedly does little to explore a program's behavior. We therefore choose the model's distribution conditioned on success as our target: discard unsuccessful outputs and preserve the relative probabilities of the successful ones.

Write $r(y_{1:H}) \in \{0, 1\}$ for the terminal reward: $r(y_{1:H}) = 1$ when the response meets the task's success criterion and $r(y_{1:H}) = 0$ otherwise. We keep this reward fixed throughout generation and training. The target is the model's distribution conditioned on success:

$$q^*(y_{1:H}) = \frac{q_{\text{ref}}(y_{1:H})r(y_{1:H})}{\alpha}, \qquad \alpha = \mathbb{E}_{y_{1:H}\sim q_{\text{ref}}}[r(y_{1:H})] > 0. \tag{1}$$

Here $\alpha = q_{\text{ref}}(G)$, where $G = \{y_{1:H} : r(y_{1:H}) = 1\}$, is the base model's probability of success, or coverage.

Generating complete responses until one succeeds samples from q^*, but requires $1/\alpha$ attempts on average. Our question is whether prefix feedback can produce this same distribution more efficiently, with the model's parameters held fixed.

1.2 The value of a prefix

A complete-response reward does not tell us which step to take next. The useful question at a prefix is: if we continue with the base model, how likely are we to succeed?

Write a response as $y_{1:H} = (y_1, \dots, y_H)$, with steps from a finite set $\mathcal{A}$. A step can be a token or a block of tokens. We use a fixed length H. Shorter responses can be padded after they end. The prefix after h steps is $y_{1:h}$, and appending the next step y_{h+1} gives $y_{1:h+1}$. Write $y_{1:0} = \varnothing$ for the empty prefix. These prefixes form a tree: the empty prefix is the root, and complete responses are leaves. Let $q_{\mathrm{ref}}(y_{1:h})$ be the probability of reaching $y_{1:h}$, so

$$q_{\mathrm{ref}}(y_{1:h+1}) = q_{\mathrm{ref}}(y_{1:h}) q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h}).$$

The **continuation value** answers the question above:

$$V(y_{1:h}) = \mathbb{E}_{q_{\mathrm{ref}}}[r(y_{1:H}) \mid y_{1:h}]. \tag{2}$$

Here the prefix $y_{1:h}$ is fixed, and the expectation averages over future steps $y_{h+1:H}$ generated by q_{ref}. This value depends on the model used to continue the response. It measures more than whether the latest step was correct: a valid step can still leave this model a difficult problem to finish.

At the root, $V(\varnothing) = \alpha$. At a leaf, the response is complete and $V(y_{1:H}) = r(y_{1:H})$. At an internal node, conditioning on the next step gives

$$V(y_{1:h}) = \sum_{y_{h+1} \in \mathcal{A}} q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h}) V(y_{1:h+1}). \tag{3}$$

Thus a prefix's value is the average of its children's values, weighted by the model's next-step probabilities. An approximate verifier $\widehat{V}$ need not satisfy this identity.

2 Sampling with exact values

Suppose we have generated a prefix $y_{1:h}$. Imagine sampling many continuations from the base model and keeping only those that succeed. Which next steps would appear among the retained responses? A step appears in proportion to two quantities: how often the model chooses it, and how often the model succeeds after taking it. These are $q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h})$ and $V(y_{1:h+1})$. Their product gives the weight of that next step among successful continuations. With exact values, we can compute these weights without generating the continuations. The following proposition gives the resulting next-step distribution.

Proposition 2.1 (Next-step probabilities of the target)**.** *For any incomplete prefix $y_{1:h}$ with $V(y_{1:h}) > 0$, the conditional distribution of the next step under q^* is*

$$q^*(y_{h+1} \mid y_{1:h}) = \frac{q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h}) V(y_{1:h+1})}{V(y_{1:h})}. \tag{4}$$

Proof. Write $q^*(y_{1:h})$ for the probability that a response drawn from q^* begins with $y_{1:h}$. Under q_{ref}, the probability of reaching this prefix and eventually succeeding is $q_{\mathrm{ref}}(y_{1:h}) V(y_{1:h})$. Conditioning on success gives

$$q^*(y_{1:h}) = \frac{q_{\mathrm{ref}}(y_{1:h}) V(y_{1:h})}{\alpha}.$$

Therefore,

$$\begin{aligned} q^*(y_{h+1} \mid y_{1:h}) &= \frac{q^*(y_{1:h+1})}{q^*(y_{1:h})} \\ &= \frac{q_{\text{ref}}(y_{1:h+1})V(y_{1:h+1})}{q_{\text{ref}}(y_{1:h})V(y_{1:h})} \\ &= \frac{q_{\text{ref}}(y_{h+1} \mid y_{1:h})V(y_{1:h+1})}{V(y_{1:h})}. \end{aligned}$$

□

To use the rule, start with the empty prefix. At each step, multiply each possible next step's model probability by its continuation value, normalize these weights, and sample one step. Append the sampled step and repeat until the response is complete. The weights sum to $V(y_{1:h})$ by (3). A zero-valued child is never chosen. Since these are the target's conditional probabilities, sequential sampling generates q^* by the chain rule. We sample according to the weights rather than always taking the highest-value child, because our goal is to preserve the relative probabilities of successful responses.

Example: excluding an invalid symbol. Consider a model that independently generates each symbol uniformly from $\{a, b, c\}$. Set $r(y_{1:H}) = \mathbf{1}\{y_{1:H} \text{ contains no } c\}$. The target is the base model conditioned on this constraint, which is uniform over $\{a, b\}^H$. Always returning a^H would satisfy the constraint, but would miss this target. We want to preserve the relative frequencies of all valid strings.

If a prefix contains c, its value is zero. Otherwise, every remaining symbol independently avoids c with probability $2/3$, so

$$V(y_{1:h}) = \left(\frac{2}{3}\right)^{H-h} \quad \text{for prefixes containing only } a, b. \tag{5}$$

The children $y_{1:h}a$ and $y_{1:h}b$ have equal value, and $y_{1:h}c$ has value zero. Here juxtaposition means appending the literal symbol to the prefix. Guided sampling therefore chooses a and b with probability $1/2$ each and never chooses c. Figure 1 shows the values and resulting response probabilities for $H = 2$.

Here $\alpha = (2/3)^H$, so complete-response rejection needs $1/\alpha = (3/2)^H$ attempts on average, each of length H. With the prefix values, guided sampling produces a valid response in H steps. This comparison assumes that the values are available and does not include the cost of obtaining them.

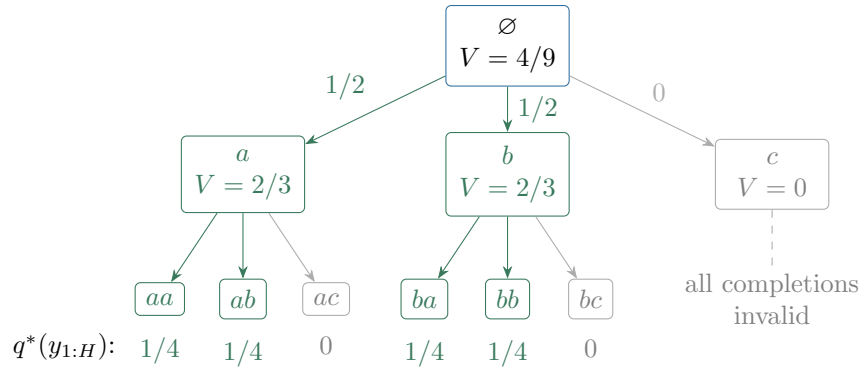


Figure 1: The two-step constraint example. The base model assigns each symbol probability 1/3. Guided sampling follows the green edges, each with probability 1/2, and gives each valid response probability 1/4. The value at a prefix is its probability of a valid completion under the base model.

3 Accumulation of verifier error

The exact sampler assumes that we know $V(y_{1:h})$ at each prefix. In the constraint example, these values had a simple formula. For an LLM, even when checking a completed answer is easy, computing the probability of a correct completion from a prefix requires averaging over the model's possible continuations. Evaluating this expectation exactly is generally too expensive. Instead, we can estimate it from sampled completions or train a verifier to predict it. Either way, we must work with approximate values.

The natural implementation substitutes nonnegative predictions $\widehat{V}$ for the exact values: reweight the next-step probabilities, normalize, sample, and append. Does this still produce approximately the right distribution? At a prefix where at least one child has positive estimated value, the approximate guided sampler uses

$$\widehat{q}(y_{h+1} \mid y_{1:h}) = \frac{q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h})\widehat{V}(y_{1:h+1})}{\sum_{z \in \mathcal{A}} q_{\mathrm{ref}}(z \mid y_{1:h})\widehat{V}(y_{1:h}z)}. \tag{6}$$

Here z ranges over possible next steps, and $y_{1:h}z$ denotes appending that step. The denominator sums over the children because the estimates need not satisfy (3). In particular, it need not equal $\widehat{V}(y_{1:h})$. Since the estimates can differ from the true continuation values, this rule need not give the conditional probabilities of q^*.

Return to the model over $\{a, b, c\}$. We keep zero values exact, so the sampler never chooses c, and use exact rewards at complete responses. Can small value errors nevertheless distort the distribution among valid responses? Suppose the verifier overestimates every incomplete valid prefix ending in a by a factor $1 + \varepsilon$:

$$\widehat{V}(y_{1:h}) = \begin{cases} (1+\varepsilon)V(y_{1:h}), & 0 < h < H \text{ and } y_h = a, \\ V(y_{1:h}), & \text{otherwise,} \end{cases} \qquad 0 < \varepsilon < 1/2.$$

Use the exact validity check at completed responses. All estimates are within a factor $1 + \varepsilon$ of their true values, but at each of the first $H - 1$ positions the sampling probabilities become

$$\widehat{q}(a \mid y_{1:h}) = \frac{1+\varepsilon}{2+\varepsilon}, \qquad \widehat{q}(b \mid y_{1:h}) = \frac{1}{2+\varepsilon}. \tag{7}$$

The last position is unbiased because its children are complete responses.

These probabilities do not depend on the earlier symbols, so the first $H-1$ choices are independent biased coin flips between a and b. Let K count the a's in these $m = H - 1$ positions. The fraction K/m is an average of independent 0/1 indicators: as the number m of positions grows, it concentrates near the probability of drawing a. Thus, by the law of large numbers, K/m converges to $1/2$ under the target distribution. Under the approximate sampler, it converges to $(1+\varepsilon)/(2+\varepsilon) > 1/2$. For fixed $\varepsilon > 0$, a threshold between these two fractions distinguishes the distributions with probability tending to one as H grows. Figure 2 illustrates this separation for a fixed verifier error. Both samplers produce only valid responses. The error is that the approximate sampler overproduces responses containing many a's. Repeating the same forward procedure gives more samples from this biased distribution.

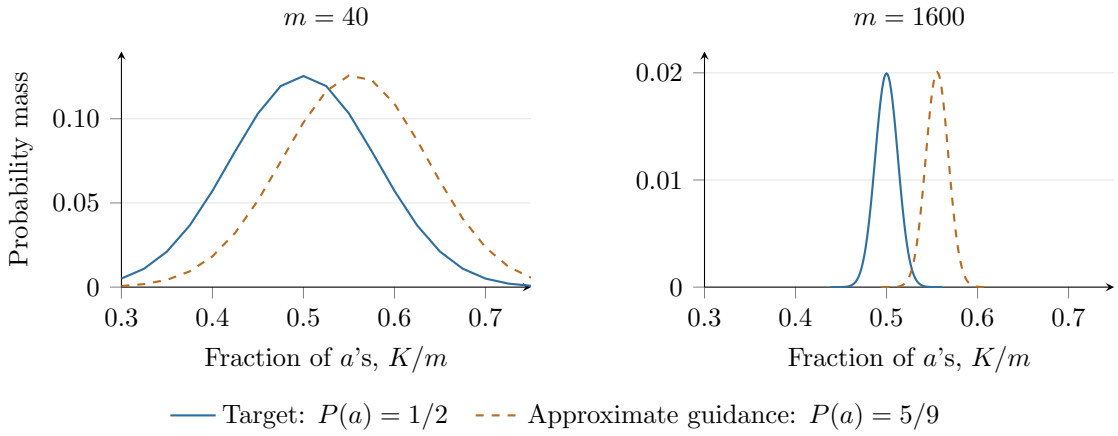


Figure 2: Binomial probabilities for the number of a's in the first $m = H - 1$ positions, with $\varepsilon = 1/4$ in both panels. The one-step probabilities differ by $1/18$. For long responses, the resulting fractions of a's are nevertheless easy to distinguish.

This example adapts the construction of Rohatgi et al. [1, Section 3], with exact terminal scores.

4 Value-guided sampling with backtracking

Can we use more computation to reduce this bias while keeping the same imperfect verifier? Return to the ABC example and focus on the next-to-last symbol. Appending a or b leaves the same true chance of success, but the inflated estimate makes the sampler choose a too often. Suppose it has just chosen a. There is now one symbol left. Appending a, b, c gives exact terminal rewards $1, 1, 0$, so the success probability from this prefix is $2/3$, below its predicted value $(1+\varepsilon)\,2/3$.

Using these exact scores to choose the final symbol avoids c, but still leaves too many responses with a in the preceding position. The feedback concerns a choice the forward sampler has already made. Erasing that a would let it reconsider the choice. The goal is to correct the frequencies of valid responses, rather than their validity. We now give a rule for making these revisions and show why it can recover the target distribution.

4.1 Appending or erasing a step

Value-guided sampling with backtracking (VGB) [1] starts at the empty response and repeatedly chooses whether to append a step or erase the last one. For example, $\varnothing \to a \to \varnothing \to b$

revises the first choice. On the prefix tree, these operations move to a child or parent. This sequence of probabilistically chosen positions is a *random walk*.

Assume that $\widehat{V}(y_{1:h}) = 0$ exactly when $V(y_{1:h}) = 0$, and omit these zero-valued branches. At complete responses, use the exact terminal score $\widehat{V}(y_{1:H}) = r(y_{1:H})$. At an incomplete, nonempty prefix $y_{1:h}$, retain the forward weight $q_{\text{ref}}(y_{h+1} \mid y_{1:h})\widehat{V}(y_{1:h+1})$ for each possible next step, and give erasing the last step weight $\widehat{V}(y_{1:h})$. This is the same score that helped the sampler enter the current prefix. It now also weights undoing that choice. Normalize all these weights together and sample a move, as shown in Figure 3. In particular, the probability of erasing is

$$\mathbb{P}(\text{backtrack} \mid y_{1:h}, \text{move}) = \frac{\widehat{V}(y_{1:h})}{\widehat{V}(y_{1:h}) + \sum_{y_{h+1} \in \mathcal{A}} q_{\text{ref}}(y_{h+1} \mid y_{1:h})\widehat{V}(y_{1:h+1})}. \tag{8}$$

The root can only append, and a complete response can only erase.

Warm-up: exact values. If $\widehat{V} = V$, the combined append weight equals the erase weight:

$$\sum_{y_{h+1}} q_{\text{ref}}(y_{h+1} \mid y_{1:h})V(y_{1:h+1}) = V(y_{1:h}).$$

At an incomplete, nonempty prefix, the sampler therefore erases with probability $1/2$ and appends with probability $1/2$. Conditional on appending, it uses exactly the guided distribution from Section 2. Exact values already permit forward sampling. The purpose of adding backtracking is to tolerate errors in the values.

With imperfect values. The combined append weight can now differ from the erase weight. At the ABC prefix above, the inflated estimate makes erasing more likely than appending. This allows the sampler to reconsider the earlier choice, but does not by itself show that repeated revisions produce the right frequencies of complete responses.

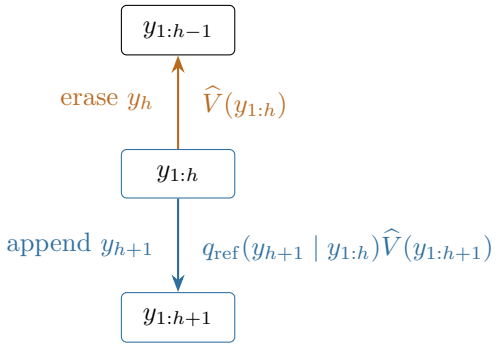


Figure 3: The erase move and one possible append move, with their unnormalized weights.

To obtain an output, run for a fixed number T of iterations and inspect the current prefix. Return it if complete. Otherwise, restart independently. Keep moving after earlier visits to leaves. With imperfect values, the first leaf can still reflect biased branch choices. We will first identify the distribution the moves preserve, then ask how long to run from the root to approach it.

4.2 The leaf distribution at equilibrium

Because the procedure can append and erase, its current position can be any retained prefix, including a complete response. Imagine many independent copies, with a given fraction at each

node. A distribution over these positions is **stationary**, or at **equilibrium**, if one more iteration leaves the node probabilities unchanged, even though individual copies move. We will find such a distribution and show that its conditional distribution over complete responses is q^*.

Balancing append and erase. Let $M(y_{1:h})$ be the sum of the available move weights, the quantity we already divide by when choosing a move. At an incomplete, nonempty prefix,

$$M(y_{1:h}) = \widehat{V}(y_{1:h}) + \sum_{y_{h+1}} q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h})\widehat{V}(y_{1:h+1}).$$

At the boundaries, include only the moves that are available:

$$M(\varnothing) = \sum_{y_1} q_{\mathrm{ref}}(y_1)\widehat{V}(y_1) \qquad \text{(append only)},$$
$$M(y_{1:H}) = r(y_{1:H}) \qquad \text{(erase only)}.$$

Consider a distribution that assigns each prefix probability proportional to $q_{\mathrm{ref}}(y_{1:h})M(y_{1:h})$. The factor M will cancel the denominator in the move probability. The factor q_{ref} will combine with the next-step probability to give the probability of the longer prefix.

Proposition 4.1 (The stationary distribution)**.** *The walk preserves the distribution*

$$\mu(y_{1:h}) = \frac{q_{\mathrm{ref}}(y_{1:h})M(y_{1:h})}{C},$$

where C makes the probabilities over all retained prefixes sum to one. Conditioned on being at a leaf, this distribution is exactly q^*.

Proof. Suppose the current position has distribution μ. Fix a prefix $y_{1:h}$ and one child $y_{1:h+1}$. We compare the probability of moving between these two prefixes in each direction.

The probability of being at the parent and appending y_{h+1} is

$$\underbrace{\frac{q_{\mathrm{ref}}(y_{1:h})M(y_{1:h})}{C}}_{\text{at the parent}} \underbrace{\frac{q_{\mathrm{ref}}(y_{h+1} \mid y_{1:h})\widehat{V}(y_{1:h+1})}{M(y_{1:h})}}_{\text{append } y_{h+1}}$$
$$= \frac{q_{\mathrm{ref}}(y_{1:h+1})\widehat{V}(y_{1:h+1})}{C}.$$

Here $M(y_{1:h})$ cancels, and the chain rule combines the two model probabilities. The probability of being at the child and erasing its last step is

$$\underbrace{\frac{q_{\mathrm{ref}}(y_{1:h+1})M(y_{1:h+1})}{C}}_{\text{at the child}} \underbrace{\frac{\widehat{V}(y_{1:h+1})}{M(y_{1:h+1})}}_{\text{erase } y_{h+1}}$$
$$= \frac{q_{\mathrm{ref}}(y_{1:h+1})\widehat{V}(y_{1:h+1})}{C}.$$

The two probabilities are equal. This holds for every parent and child, including connections to the root and leaves. Summing over a prefix's parent and children, its total incoming probability equals its total outgoing probability. Every run makes a move, so the outgoing probability is its current probability $\mu(y_{1:h})$. The same probability therefore arrives after the move, proving stationarity.

At a complete response, $M(y_{1:H}) = r(y_{1:H})$, so $\mu(y_{1:H}) = q_{\text{ref}}(y_{1:H})r(y_{1:H})/C$. Conditioning on a leaf cancels the common C:

$$\begin{aligned}\mu(y_{1:H} \mid \text{leaf}) &= \frac{q_{\text{ref}}(y_{1:H})r(y_{1:H})}{\sum_{z_{1:H}} q_{\text{ref}}(z_{1:H})r(z_{1:H})} \\ &= \frac{q_{\text{ref}}(y_{1:H})r(y_{1:H})}{\alpha} = q^*(y_{1:H}).\end{aligned}$$

The sum ranges over complete responses. Zero-valued leaves contribute zero and may be included in this sum. □

The internal estimates affect how often prefixes are visited, but the relative weights of complete responses depend only on their terminal scores. Thus, in the ABC example, the equilibrium leaf distribution is uniform over $\{a, b\}^H$ despite the overestimates at prefixes ending in a. Exact terminal scores matter: replacing r by $\widehat{r}$ would instead give leaf probabilities proportional to $q_{\text{ref}}(y_{1:H})\widehat{r}(y_{1:H})$.

4.3 How running longer reduces the sampling error

The proof shows that equilibrium is preserved if the current position already has distribution μ. Our algorithm starts at the root, with all probability at that one node. *Convergence* means that the distribution from this initialization approaches μ as the number of iterations grows. The exact-value case lets us see what must change during a run.

Convergence with exact values. Recall the warm-up with $\widehat{V} = V$. The depth moves left or right with equal probability at interior positions, and inward at the endpoints:

$$\underset{\text{root}}{0} \longleftrightarrow 1 \longleftrightarrow \cdots \longleftrightarrow H-1 \longleftrightarrow \underset{\text{complete}}{H}$$

Starting from the root, conditional on having length h at any fixed time when that length can occur, the prefix has the distribution of the first h steps of a response from q^*. To see why, note that the choice to append or erase depends only on depth, so it does not change the relative frequencies of prefixes at that depth. Appending uses the target's next-step conditional. Erasing the last step of the first $h+1$ steps of a response from q^* leaves its first h steps. Both ways of arriving at length h therefore give the same prefix distribution. This property holds initially at the root and is preserved at each iteration.

Only the distribution over depths needs to settle. In particular, the conditional leaf distribution is already correct whenever a leaf can occur. To obtain convergence, add a pause with probability $1/2$ at each iteration, and otherwise make an append/erase move. Without pauses, the depth alternates between even and odd, so its distribution cannot settle. Pausing preserves both μ and the conditional prefix distributions.

For intuition about the time scale, consider the same left/right moves without endpoints or pauses. The sum of t independent, equally likely $+1/-1$ moves has mean zero and variance t. Its standard deviation is $\sqrt{t}$, so a typical displacement of order H suggests a time scale of H^2. The convergence theorem below makes this intuition precise for the actual walk, with additional logarithmic accuracy factors.

What changes with imperfect values. Now the choice of move can depend on the particular prefix, and appending can bias the frequencies of continuations. Convergence must correct these frequencies as well as the distribution over depths. At equilibrium, the leaf distribution is still q^*. The remaining question is how quickly the prefix distribution approaches equilibrium from the root. The paper compares the probability a branch has at equilibrium with the probability leaving it in one iteration. A uniform bound on relative value errors keeps the latter from being too small compared with the former, preventing long delays in revising earlier choices [1, Appendix I.3].

Write $Y^{(t)}$ for the response prefix after t iterations, including pauses. Here t counts iterations, while h denotes prefix length. Start at $Y^{(0)} = \varnothing$ and use the earlier output rule after T iterations. Convergence controls this fixed-time distribution, which is why the output rule inspects the prefix at time T.

The following result of Rohatgi et al. [1, Theorem 4.2 and Appendix I.2] bounds the required number of iterations and ensures that a leaf is observed often enough. Here κ describes the verifier's accuracy, while δ is the desired accuracy of the returned response distribution. We state the result without proof.

Theorem 4.2 (Convergence of VGB)**.** *Suppose the terminal scores are exact and, for every incomplete prefix,*

$$\kappa^{-1} V(y_{1:h}) \le \widehat{V}(y_{1:h}) \le \kappa V(y_{1:h}), \qquad \kappa \ge 1.$$

There is a universal constant C_0 such that, for any $0 < \delta < 1/2$, running VGB for

$$T \ge C_0 \kappa^4 H^2 \log\left(\frac{\kappa H}{\delta}\right)$$

iterations gives a leaf $Y^{(T)}$ with probability at least $1/(8\kappa H)$. Conditioned on being a leaf, its distribution is within total variation δ of q^*.

Total variation at most δ means that the probabilities assigned to any set of responses differ by at most δ. The independent restart rule returns exactly the conditional distribution in the theorem. The lower bound on observing a leaf ensures that we need at most $8\kappa H$ runs on average before returning a response. For fixed κ and δ, the expected number of walk iterations is therefore polynomial in H, whereas complete-response rejection can require exponentially many attempts. Each walk iteration may require scoring several candidate next steps.

κ can stay fixed as δ decreases: more computation can reduce the sampling error without improving the verifier. This requires relative accuracy at *every* prefix, including correct zero estimates. Small average prediction error alone does not imply this assumption.

Connection to approximate counting. For a uniform base model and binary rewards, $V(y_{1:h})$ is the fraction of valid completions. Estimating this fraction is equivalent to estimating their number. In this setting VGB reduces to the Sinclair–Jerrum walk [2], which uses approximate counts to sample nearly uniformly. VGB extends this method to nonuniform language models.

Sampling experiments. Rohatgi et al. [1] test the same ABC constraint task with a verifier trained from sampled continuations. Their experiments stop at the first leaf to reduce computation, rather than using the fixed-time observation rule of the theorem. Figure 4 compares VGB with the forward sampler. Both use exact terminal checks and return only valid responses, yet VGB's output distribution is closer to the target. The difference becomes more pronounced for longer responses.

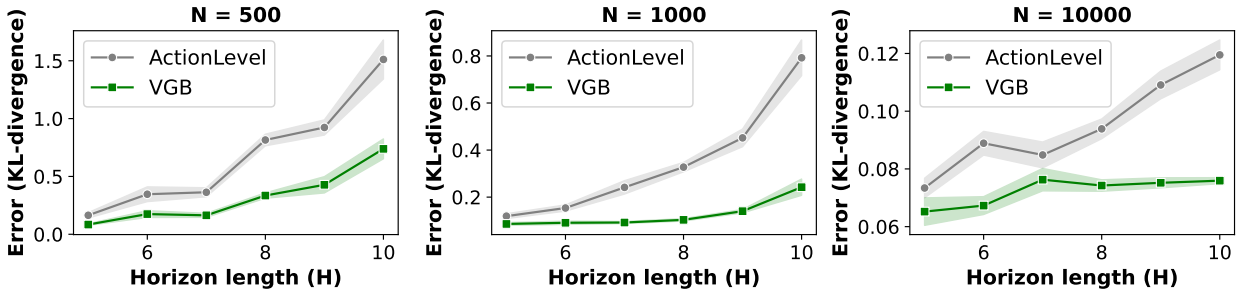


Figure 4: ABC experiment from Rohatgi et al. [1, Figure 3]. Each panel uses N verifier-training samples at each prefix length. The horizontal axis is response length H, and the vertical axis is estimated KL divergence from the output distribution to the target (lower is better, and scales differ across panels). "ActionLevel" is the forward sampler, and VGB adds backtracking. Curves show means over ten repetitions, and shaded bands show standard errors.

Language generation. The paper also uses Qwen2.5-0.5B to generate 32-token English responses without the letter "e." The forward sampler filters out forbidden next tokens. VGB can also revise earlier choices. Here the prefix scores use a simple heuristic based on whether the constraint still holds, rather than a trained value predictor. In Figure 5, VGB wins about 55% of coherence comparisons at several backtracking settings. This measures writing quality under a constraint, not closeness to the target distribution.

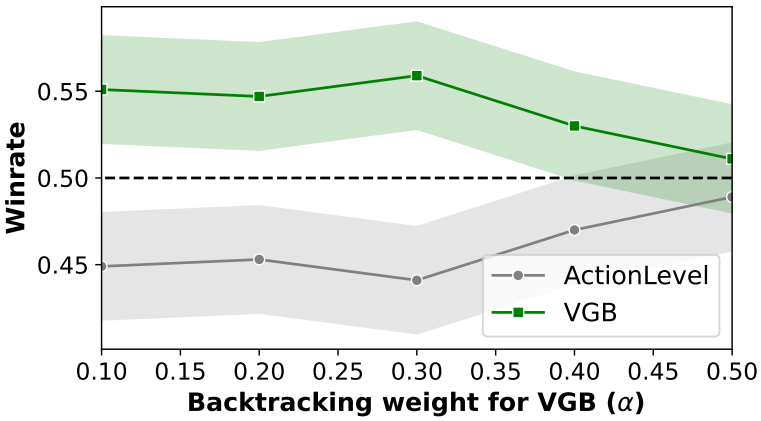


Figure 5: Letter-avoidance experiment from Rohatgi et al. [1, Figure 4, left]. GPT-4o-mini judges coherence in 1000 paired comparisons per setting. The curves give each method's win rate against the other. A win rate of 50% means equal preference. The plot uses the paper's symbol α for a backtracking multiplier, distinct from our success probability. "ActionLevel" is the forward sampler. Shaded bands are 95% binomial confidence intervals.

5 Training with process values

So far, we kept q_{ref} fixed and changed how we generated responses from it. We now train the model, using the expected final reward objective from the previous lecture on policy gradients. Responses are sampled directly from the current model q_θ for the update.

One completed response is noisy feedback about an earlier step. In the uniform ABC model, aa and ac start with the same a, but their final rewards are 1 and 0. The difference comes from the second step. A continuation value averages over these possible futures, making the feedback for the first step less dependent on which continuation happened to be sampled. We will keep the same expected gradient while changing the information used to estimate it.

Fix a prompt and keep its binary terminal reward $r(y_{1:H})$ fixed as the model changes. The objective is

$$J(\theta) = \mathbb{E}_{y_{1:H}\sim q_\theta}[r(y_{1:H})].$$

At step h, the prefix before choosing y_h is $y_{1:h-1}$. Write

$$g_h = \nabla_\theta \log q_\theta(y_h \mid y_{1:h-1})$$

for the likelihood gradient of step h. Recall the REINFORCE identity:

$$\nabla_\theta J(\theta) = \mathbb{E}_{y_{1:H}\sim q_\theta}\left[\sum_{h=1}^{H} r(y_{1:H}) g_h\right].$$

A baseline depending only on the prompt changes this shared weight but still does not distinguish the steps.

Averaging over possible continuations. Define the continuation value under the current model:

$$V_\theta(y_{1:h}) = \mathbb{E}_{q_\theta}[r(y_{1:H}) \mid y_{1:h}].$$

The expectation fixes $y_{1:h}$ and averages over future steps under q_θ. After taking step y_h, the probability of eventual success is $V_\theta(y_{1:h})$. Before that step, it is $V_\theta(y_{1:h-1})$. At a complete response, $V_\theta(y_{1:H}) = r(y_{1:H})$.

Group responses by their prefix $y_{1:h-1}$ and chosen step y_h. Within each group, g_h is the same. Only the continuation and its reward vary. We can average the reward within each group first, then average across groups according to their sampling probabilities. This is the law of total expectation:

$$\begin{aligned}\mathbb{E}[r(y_{1:H})g_h] &= \mathbb{E}\big[\mathbb{E}[r(y_{1:H}) \mid y_{1:h}]g_h\big] \\ &= \mathbb{E}[V_\theta(y_{1:h})g_h].\end{aligned}$$

The equality preserves the mean, but the estimates on individual responses can differ. After the first a in our example, the equally likely completions aa, ab, ac have rewards $1, 1, 0$. Replacing this random reward by $V_\theta(a) = 2/3$ keeps its conditional mean while removing the variation due to the second step. In general, exact values average away continuation noise from each step's gradient contribution. This can make a finite batch more informative while preserving the expected gradient.

Comparing a step with the alternatives. To judge the chosen step, compare its value with the average over the model's possible next steps. This average is $V_\theta(y_{1:h-1})$.

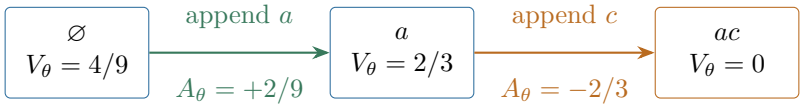
We now apply the baseline argument from the previous lecture to each prefix. Conditional on $y_{1:h-1}$, the baseline $V_\theta(y_{1:h-1})$ does not depend on which next step is sampled. The next-step probabilities sum to one, so the same log-derivative identity gives $\mathbb{E}[g_h \mid y_{1:h-1}] = 0$. Subtracting the baseline therefore leaves the expected gradient unchanged. The resulting weight is the step's **advantage**:

$$A_\theta(y_{1:h}) = V_\theta(y_{1:h}) - V_\theta(y_{1:h-1}). \tag{9}$$

The advantage is positive when the chosen step raises the probability of eventual success relative to the prefix average, and negative when it lowers that probability. Combining the two changes gives

$$\boxed{\nabla_\theta J(\theta) = \mathbb{E}_{y_{1:H}\sim q_\theta}\left[\sum_{h=1}^{H} A_\theta(y_{1:h}) g_h\right].} \tag{10}$$

Figure 6 applies these weights to the sampled response ac in our two-step constraint task, with q_θ uniform over $\{a, b, c\}$ at each step.



Outcome-only weights: 0 for both steps.

Figure 6: The sampled response ac fails, but its first step increases the probability of success. Advantages give positive credit to that step and negative credit to the next. REINFORCE using only the final reward, without a baseline, gives both steps weight zero.

The identities above use exact values under the current model. As before, hold the sampled text and numerical advantage estimates fixed when differentiating the log probabilities. In practice, we must estimate the values. The quality and cost of these estimates determine whether the extra feedback helps training.

6 Obtaining process scores

Continuations and outcome checks. From a prefix $y_{1:h}$, sample several independent completions and average their terminal rewards $r(y_{1:H})$. Using q_{ref} for the completions estimates $V(y_{1:h})$. Using the current model q_θ estimates $V_\theta(y_{1:h})$. The resulting pairs (prefix, estimated value) can also be used to train a value predictor, so that future scores need not be obtained by fresh rollouts. Finite samples can miss rare successful completions, so these estimates do not automatically satisfy Theorem 4.2's uniform relative-error assumption.

For an unbiased estimate of the gradient in (10), use the current model for both the sampled response and its continuation rollouts. Estimate the prefix baseline from separate rollouts, so it does not depend on the next step sampled for the update. Learned values after the chosen step, or such values reused after policy updates, need not give an unbiased gradient.

VinePPO [3] uses continuation rollouts to estimate values and compute step-level advantages for LLM training. In its MATH and GSM8K experiments, it reached PPO's best accuracy in less wall-clock time, despite the extra continuations. This is an empirical benefit, not a general runtime guarantee.

Judgments of step correctness. Human annotators or an AI judge can instead evaluate each reasoning step in context, and a verifier can be trained to predict those labels. Lightman et al. [4] used human annotations to construct PRM800K and train process verifiers for selecting mathematical solutions. A model trained on these labels predicts the correctness of a step, rather than the probability that a particular model will complete the response successfully.

Step-correctness scores can also define a training reward, such as their sum over a response. This chooses an objective based on the scores, whereas the continuation values above preserve

the expected final-reward gradient. In PURE [5], summing step scores encouraged responses that accumulated high scores without finishing the solution. Using the minimum score from the current step onward for credit assignment delayed reward hacking, and adding final-answer checks improved stability further in their experiments. A score useful for ranking responses need not define a good training objective.

References

[1] Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. *Taming Imperfect Process Verifiers: A Sampling Perspective on Backtracking.* 2025. arXiv:2510.03149.

[2] Alistair Sinclair and Mark Jerrum. *Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains.* Information and Computation, 82(1):93–133, 1989. Article.

[3] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. *VinePPO: Refining Credit Assignment in RL Training of LLMs.* ICML, 2025. Proceedings.

[4] Hunter Lightman et al. *Let's Verify Step by Step.* ICLR, 2024. arXiv:2305.20050.

[5] Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. *Stop Summation: Min-Form Credit Assignment Is All Process Reward Model Needs for Reasoning.* NeurIPS, 2025. arXiv:2504.15275.