Layer Verification Accelerates Speculative Tree Decoding

Jaeyoung Cha [*1] **Hanseul Cho** [*1] **Chulhee Yun** [1]

Abstract

Autoregressive decoding, a major bottleneck in LLM inference, requires a complete forward pass for each token generation. Speculative decoding mitigates this cost by using a fast draft model to propose multiple candidate tokens and a verifier to accept or correct them, while preserving the target distribution. Token Verification (TV), a common baseline verification method for multi-step drafts, repeatedly applies a single-step verification rule in the sequence direction and stops at the first rejection. However, it is suboptimal even for drafts with a single candidate per step. We propose **Layer Verification (LV)**, a new modular lifting strategy that converts any single-step verification rule into a draft-tree verifier, improving acceptance efficiency over TV. LV assigns appropriate scores to each tree node according to a single-step verification rule, coordinating acceptance mass across nodes within each layer. It then sweeps the draft tree layer by layer, selecting the accepted endpoint and sampling the corrected token. We prove that LV is lossless and preserves the target distribution when the underlying single-step verifier is lossless. Notably, when instantiated with Global Resolution (Thomas & Pal, 2026b), LV attains near-optimal expected acceptance length in both single-step multi-candidate and multi-step single-candidate drafts. Experiments on synthetic autoregressive models and LLM decoding corroborate the efficacy and efficiency of our method.

1. Introduction

Autoregressive large language models (LLMs) are deployed across a wide range of applications. However, their inference efficiency is often hindered by the need to perform as many forward passes as the number of generated tokens, limiting their usability and scalability. Speculative decoding (Stern et al., 2018; Xia et al., 2023; Leviathan et al., 2023) mitigates this bottleneck by using a fast draft model to propose candidate tokens for multiple next steps and a verifier (an LLM with a draft verification rule or the rule itself) to accept or correct those proposals in parallel, while matching the target distribution. This approach allows multiple tokens to be generated with a single call to the target model, reducing latency without sacrificing generation quality. See Xia et al. (2024); Hu et al. (2025a) for more details.

Recent work has made substantial progress on the former component, via training a stronger draft model and improving the construction of draft trees (Cai et al., 2024; Chen et al., 2024; Li et al., 2024a;b; 2025; Wang et al., 2025). In this paper, in contrast, we focus on the latter component: the *verification rule design* (cf. Fig. 1). Given a draft tree and draft/target conditional probability masses for each tree node, designing a verification rule becomes a mathematical problem closely related to optimal transport: how should one maximize acceptance efficiency while ensuring the final output distribution matches the target distribution?

Prior verifier designs mostly focus on the *single-step* setting, concerning the next-token candidates. Representative examples include speculative sampling (SpS) (Leviathan et al., 2023), k-sequential selection (K-SEQ) (Sun et al., 2023), recursive rejection sampling (RRS) (Jeon et al., 2024; Yang et al., 2025), and Global Resolution (GR) (Thomas & Pal, 2026b). When such single-step verifiers are applied to *chain-/tree-structured* drafts, the standard approach is to repeatedly invoke them token by token in the forward sequence direction. We call this approach *Token Verification* (TV).[1] In this sense, TV is not a specific verifier by itself, but rather a meta-algorithm that lifts a single-step verifier to multi-step drafts, including both chains and trees. Here, we use the term "lifting strategy" to refer to such a meta-algorithm.

Despite its popularity, TV has a clear weakness. It discards subsequent draft tokens after a rejection, so it cannot account for the joint acceptance probability for the entire decoded sequence. For this, especially for single-chain drafts, TV has been shown to be suboptimal in terms of decoding efficiency (Sun et al., 2025). These limitations are exacer-

[1]Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST AI). Correspondence to: Chulhee Yun <chulhee.yun@kaist.ac.kr>.

Accepted at *Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM) of 43^{rd} International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026.

[1]This is also called *SpecTr* or DraftSelection in Sun et al. (2023). We bring the name TV from Sun et al. (2025).

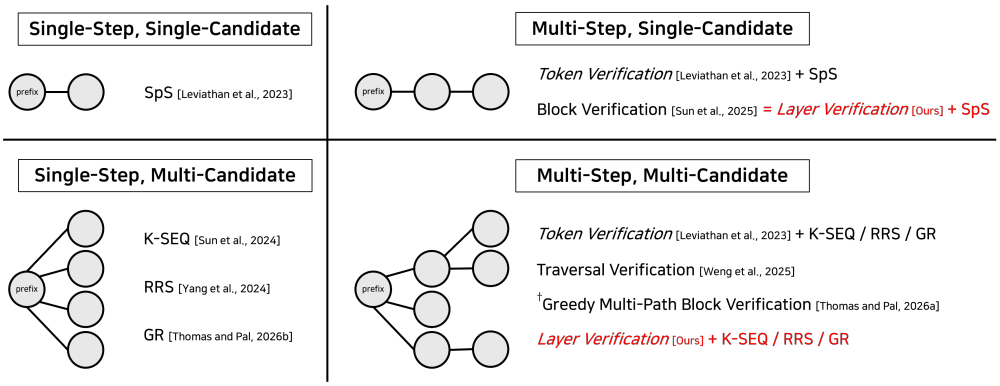


Figure 1. Overview of verification rules for speculative (tree) decoding. *Italicized* entries denote lifting strategies that extend single-step verifiers to multi-step drafts. In particular, LV-lifted SPS recovers BV in the single-chain case, while LV-lifted K-SEQ, RRS, and GR yield tree verifiers in the multi-candidate setting. †Thomas & Pal (2026a) design their algorithm specifically for multi-chain drafts rather than general tree shapes.

bated for tree-shaped drafts, where evaluating the full tree could allow longer accepted paths per target-model call.

To address this issue, we introduce a new lifting strategy, *Layer Verification* (LV), for speculative tree decoding. Each iteration of LV decomposes into two stages: the *forward node-scoring* stage and the backward sampling stage. In the first stage, LV assigns each draft tree node an appropriate score through a forward pass, interpreted as the probability that the final accepted path passes through the node. We design a computationally feasible scoring rule by extending the flow-network perspective on verifiers (Sun et al., 2024; Thomas & Pal, 2026b) to multi-step setups, efficiently distributing mass across competing branches (Sec. 3). In the second stage, LV converts these scores into a lossless verification rule. It operates by sweeping the draft tree layer by layer from the deepest layer, deciding whether a node in the current layer should be the endpoint of an accepted path; the corrected token is then sampled from a resampling distribution computed at that endpoint.

We prove that LV preserves the target distribution whenever instantiated with a lossless single-step verifier. Moreover, our experiments corroborate that LV-lifted verifiers consistently outperform their TV counterparts across tree structures and benchmarks in both block efficiency (i.e., generated sequence length per target-model forward pass) and throughput (tokens/sec).

Our contributions are summarized as follows:

- We propose **Layer Verification (LV)**, a novel two-stage meta-algorithm that converts any single-step verifier into a multi-step verifier for tree-shaped drafts. (Sec. 4).
- We prove that LV is lossless whenever the underlying single-step verifier is lossless (Thm. 4.3). Notably, when equipped with GR, LV yields a (near)-optimal block efficiency for single-step multi-candidate & multi-step single-candidate drafts. To our knowledge, this is the first lossless verification rule to achieve optimality in both settings.
- We numerically evaluate LV on controlled toy setups and LLM decoding (Sec. 5). We mainly compare LV-lifted single-step verifiers (among $\{$K-SEQ, RRS, GR$\}$) with their TV counterparts. The toy experiment validates both the losslessness of LV and its ability to accept more draft tokens than TV. Real-world experiment on SpecBench (Xia et al., 2024) corroborates the benefits of LV in both block efficiency and throughput.

1.1. Significance of Studying a Lifting Strategy

We explore the 'lifting strategy' as a new design axis in speculative decoding, an underexplored direction in most prior works. We ask whether a better lifting strategy can accept more tokens per target-model call while preserving the target distribution, given the same single-step verifier.

Notably, TV can be strictly suboptimal in multi-step settings. Sun et al. (2025) demonstrate this in the simplest multi-step setting: single-chain drafts with one candidate token per step. They propose *Block Verification* (BV), an optimal single-chain verifier for block efficiency, verifying an entire draft block at once by considering the joint distribution, rather than verifying tokens step by step. This raises the question of whether we can improve TV in multi-step, multi-candidate settings, although finding an ultimately optimal verifier is combinatorially difficult.

We highlight two major recent works studying multi-step verifiers. Traversal Verification (TRV) (Weng et al., 2025) extends RRS to tree drafts via a tree traversal and subsumes BV for single-chain drafts. However, it does not generically incorporate other single-step verifiers; thus, beating RRS is difficult even in the single-step setting. Greedy Multi-Path Block Verification (GBV) (Thomas & Pal, 2026a) selects one of multiple draft chains and applies BV, but it is not immediately applicable to general trees with branching at multiple depths. However, a lifting strategy such as LV separates local verifier design from tree-level coordination, thereby making the problem more tractable and modular.

A successful lifting strategy would amplify the value of prior advances in single-step verification. The modular frame-

work allows existing single-step verifiers to be reused in a stronger multi-step verifier, eliminating the need for redesign from scratch. Thus, a single improvement to the lifting strategy can enhance many existing verifiers, and future single-step verification advances can be easily integrated into the tree-level lifting framework.

2. Overview of Problem Settings

Vocabulary and Strings. The vocabulary Σ is a finite set of tokens. A string over Σ is a finite sequence of tokens in Σ. We denote the set of all finite-length strings by Σ^*. For a string $s_1 s_2 \cdots \in \Sigma^*$ and a sequence of indices J, we often use the J-substring as $s_J = (s_j)_{j \in J}$.

Draft/Target Models & Distributions. We view the draft model $M_s \sim (p, \mathcal{T})$ as a tree generator equipped with a base draft distribution p and a *tree configuration* $\mathcal{T} = (V, E)$.[2] Denote the set of depth-t nodes by $\mathcal{L}_t$ $(t = 0, \ldots, H)$; the path from root to a node v by $\mathsf{Path}[v]$; child nodes by $\mathsf{Ch}[v]$; parent node by $\mathsf{par}[v]$; and immediate siblings (including itself) by $\mathsf{Sib}[v] := \mathsf{Ch}[\mathsf{par}[v]]$. Denote by p_v the draft conditional probability given draft prefix $X_{\mathsf{Path}[v]}$ (of non-leaf v). When M_s receives a prefix $x_0 \in \Sigma^*$ as an input, it returns a draft tree whose node $v \in V$ contains a candidate symbol $x_v \in \Sigma$, as well as $p_v(\cdot)$ if v is non-leaf. Conversely, the target model $M_b \sim q$ receives the draft tree and assigns each node v a conditional target probability $q_v(\cdot)$ in parallel.

Speculative Tree Decoding (Alg. 3). We focus on the class of verifiers that take $\mathcal{T}$ (defined above) as input and return a decoded sequence (i.e., the accepted draft token path + a corrected token at the end), without modifying the target distribution in subsequent verification rounds.

We refer readers to Appx. D for a more detailed description of our problem setting and relevant discussions.

3. Flow-Network Perspective on Verifiers

We revisit verification rules from the perspective of flow networks. Its implications for multi-step verification are the main motivation for our LV, as will be introduced in Sec. 4 Refer to Appx. C for a brief introduction to flow networks and the relevant notation. Throughout this section, drafts are sampled with a fixed prefix x_0.

3.1. Single-Step Drafts

Consider a single drafted token $X_1 \sim p(\cdot \mid x_0)$. If a verifier accepts a token $x \in \Sigma$ with conditional probability $\mathsf{a}_1(x)$ given $X_1 = x$, then the unconditional accepted mass at x is

$$f(x \to x) := \Pr[X_1 = x \text{ and } X_1 \text{ is accepted}] = p(x)\mathsf{a}_1(x).$$

Any lossless verifier should meet two capacity constraints: $f(x \to x) \le p(x)$ since x can be accepted only after drafted, and $f(x \to x) \le q(x)$ since the final output cannot assign more mass to x than the target distribution does, to prevent overestimation and guarantee losslessness. Thus, we have $f(x \to x) \le \min\{p(x), q(x)\}$ $(\forall x \in \Sigma)$.

Fig. 2a depicts this constraint formulated as a flow network from source s to sink t. Conversely, any feasible flow f defines a verifier by accepting $X_1 = x$ with conditional probability $\mathsf{a}_1(x) = f(x \to x)/p(x)$ for every $x \in \Sigma$. The gap between the target-to-sink flow $f(x \to \mathsf{t})$ and the sink-side capacity $q(x)$ naturally defines the resampling distribution of corrected token, thereby achieving losslessness. Moreover, the maximum acceptance mass at x is $\min\{p(x), q(x)\}$, offering the optimal acceptance mass $\mathsf{a}_1^\star(x) = \min\{p(x), q(x)\}/p(x) = \min\{1, q(x)/p(x)\}$, which recovers SpS (Leviathan et al., 2023).

Single-step Multi-Candidate Drafts. We can construct a similar flow network (with more source-side nodes) to analyze the case of proposing multiple next-token candidates: see Fig. 2b. We defer its detailed description to Appx. C.1.

3.2. Multi-Step Drafts

Let us consider the simplest multi-step setting: a single chain. Consider a draft sequence $X_{1:H}$ of length H. The verifier accepts a prefix $X_{1:t}$ for some $t = 0, \ldots, H$. First, let us see how a lossless multi-step verification rule induces recursive flow-network-type constraints on probabilities, and vice versa. For $1 \le t \le H$, define conditional and t-unconditional acceptance mass given $x_{<t}$ as

$$\mathsf{a}_t(x_t \mid x_{<t}) = \Pr[\text{Accept } X_t = x_t \mid x_0, X_{\le t} = x_{\le t}],$$
$$f_t(x_t \to x_t \mid x_{<t}) = \Pr[\text{Accept } X_t = x_t \mid x_0, X_{<t} = x_{<t}].$$

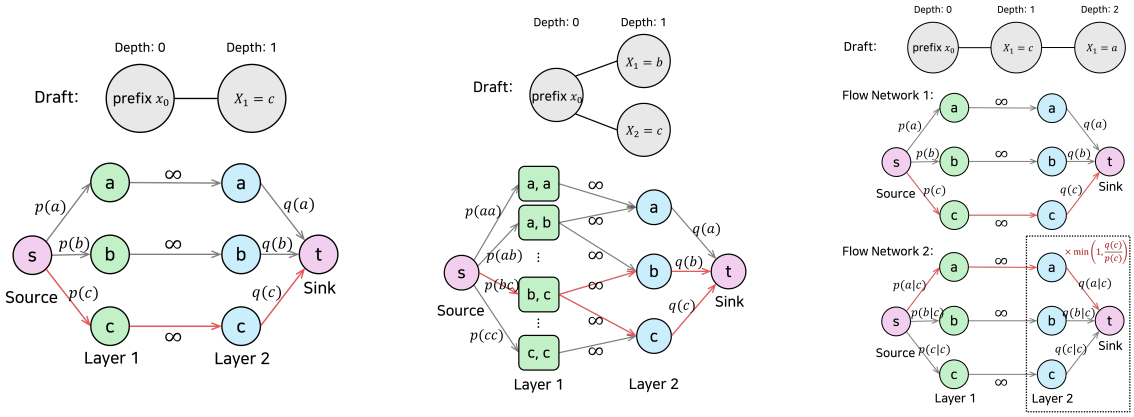
Note that $f_t(x_t \to x_t \mid x_{<t}) = p(x_t \mid x_{<t})\mathsf{a}_t(x_t \mid x_{<t})$ represents the probability of accepting at least t tokens in a row, given $x_{<t}$. From now on, let us omit the conditions if clear from the context; let $p_t(\cdot) = p(\cdot \mid x_{\le t})$ and similarly $q_t(\cdot)$.

At $t = 1$, the same set of constraints applies as in Sec. 3.1: $f_1(x \to x) \le \min\{p_0(x), q_0(x)\}$. In general $(t \to t+1)$, we have two capacity constraints: $f_{t+1}(x \to x) \le p_t(x)$ since x can be accepted only after drafted, and $f_{t+1}(x \to x) \le q_t(x)\mathsf{a}_t(x_t)$ since the target mass $q_t(x)$ can contribute only when the acceptance at step t is guaranteed $(\forall x \in \Sigma)$. These constraints across H steps naturally translate into H flow networks, each with flows defined by f_t, as depicted in Fig. 2c. Notably, the target-side capacities are "scaled" by $\mathsf{a}_t(x_t)$, naively comparing to the single-step case.

Next, we explain how maximizing the expected acceptance length t_{out} relates to solving max-flow problems for all flow networks $f_t(\ldots)$ jointly. Observe that

$$\begin{aligned}\mathbb{E}[t_{\mathsf{out}}] &= \textstyle\sum_{t=1}^{H} \Pr[t_{\mathsf{out}} \ge t] \quad \because \text{Summation by parts} \\ &= \textstyle\sum_{t=1}^{H} \sum_{x_{\le t}} \Pr[t_{\mathsf{out}} \ge t, X_{\le t} = x_{\le t}] \\ &= \textstyle\sum_{t=1}^{H} \sum_{x_{<t}} p(x_{<t}) \cdot \sum_{x_t} f_t(x_t \to x_t \mid x_{<t}),\end{aligned}$$

[2] We focus on a fixed tree configuration and i.i.d. with-replacement sampling of candidate tokens for child nodes at each tree node, unless specified otherwise.



(a) Single-step single-candidate. *(b)* Single-step double-candidate. *(c)* Two-step single-candidate.

Figure 2. Flow-network visualizations of verification rules ($\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$) The edge labels indicate their capacities.

and it is a (non-negatively) weighted sum of values of all flow networks we consider. In the current setting, a greedy sequential maximization ($t = 1, \dots, H$) yields the joint maximum flows, which offers with $\mathsf{a}^\star_0 := 1$ that

$$\mathsf{a}^\star_{t+1}(x_{t+1}) = \min\{1,\ \mathsf{a}^\star_t(x_t) \cdot q_t(x_t) \,/\, p_t(x_t)\} \quad (t \geq 0).$$

It precisely recovers BV (Sun et al., 2025).

4. Layer Verification

The chain example in Sec. 3.2 suggests tracking the probability that the accepted draft path includes a node; we call this the node's *inclusion score*. LV leverages this idea to design a draft-tree verifier in two stages. First, a *forward node-scoring stage* assigns inclusion scores layerwise by transforming a single-step verifier into a feasible flow network. Second, a *backward sampling stage* determines the endpoint of the accepted path from the deepest layer of the draft tree and samples a corrected token from a residual distribution. We begin with the latter stage, as it provides a high-level skeleton of LV: as long as the first stage is performed correctly, we can derive the endpoint selection and residual correction rules that guarantee that LV is lossless.

4.1. Backward Sampling Stage for Losslessness

We here describe the main loop of LV that determines an accepted draft token path and samples a correction token, as standard verification rules do.

Suppose we have an inclusion score precomputed for each node, as defined below.

Definition 4.1 (Inclusion Score)**.** For a node $v \in \mathcal{L}_t$, its **inclusion score** $\mathsf{a}_v\big(X_{\mathsf{Path}[v]}, X_{\mathsf{Sib}[v]}\big)$ (or, a_v in short) is to be determined by the probability of accepting all nodes in $\mathsf{Path}[v]$, given draft tokens at $\mathsf{Path}[v] \cup \mathsf{Sib}[v]$. In particular, $\sum_{v \in \mathcal{L}_t} \mathsf{a}_v = \Pr\left[t_{\mathsf{out}} \geq t \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \leq 1$; we let $\mathsf{a}_0 = 1$.

It implicitly assumes the independence of the inclusion score from the draft tokens outside $\mathsf{Path}[v] \cup \mathsf{Sib}[v]$. We can consider a more general form that also yields an essentially identical proof of our LV's losslessness. However, we choose this definition here for practical convenience in later descriptions, allowing us to consider a small-scale flow network per node and its draft tokens, converted from a single-step verifier. In particular, the quantity

$$f_v\big(x_{\mathsf{Ch}[v]} \to \tilde{x} \,\big|\, X_{\mathsf{Path}[v]}\big) := \sum_{w \in \mathsf{Ch}[v]} \mathsf{a}_w\big(X_{\mathsf{Path}[v]}, x_{\mathsf{Ch}[v]}\big)\ p_v(x_{\mathsf{Ch}[v]})\ \mathbb{1}[x_w = \tilde{x}]$$

can be interpreted as a *flow* from a tuple node $x_{\mathsf{Ch}[v]}$ to a target token $\tilde{x}$ in the flow network associated with the node v, since it represents the probability of accepting a node w with token $x_w = \tilde{x}$ given its ancestors' draft tokens. In addition, let us define a useful quantity called *outgoing flow* at target token $\tilde{x}$ as $\mathsf{flow}_v(\tilde{x}) := \sum_{x_{\mathsf{Ch}[v]}} f_v(x_{\mathsf{Ch}[v]} \to \tilde{x})$.

To design LV as a lossless verification rule, we introduce the following assumption on the inclusion scores, which directly extends the flow-network analogy described in Sec. 3.

Assumption 4.2. Fix a depth $t < H$, node $v \in \mathcal{L}_t$, and $\tilde{x} \in \Sigma$. Then, given draft tokens up to depth $t+1$, the inclusion scores (Defn. 4.1) must satisfy $\mathsf{flow}_v(\tilde{x}) \leq q_v(\tilde{x}) \cdot \mathsf{a}_v$.

We note that any lossless verifier satisfies the assumption above, as it is necessary to have valid formulas of an *acceptance rate* (i.e., probability of selecting v as the accepted endpoint) and a residual distribution at node v ($\in \mathcal{L}_t$):

$$\mathsf{h}_v := \Pr\left[v_{\mathsf{out}} = v \,|\, t_{\mathsf{out}} \leq t\right] = \frac{\mathsf{a}_v - \sum_{x' \in \Sigma} \mathsf{flow}_v(x')}{1 - \sum_{v' \in \mathcal{L}_t} \sum_{x' \in \Sigma} \mathsf{flow}_{v'}(x')}, \tag{1}$$

$$p^{\mathsf{res}}_v(\tilde{x}) := \Pr\left[Y = \tilde{x} \,|\, v_{\mathsf{out}} = v\right] = \frac{q_v(\tilde{x}) \cdot \mathsf{a}_v - \mathsf{flow}_v(\tilde{x})}{\mathsf{a}_v - \sum_{x' \in \Sigma} \mathsf{flow}_v(x')}, \tag{2}$$

omitting the conditioning on draft tokens.

Based on these ingredients, we now present the bottom-up selection stage in Alg. 2. We also present our theoretical result for the losslessness of LV.

Theorem 4.3 (Losslessness of LV)**.** *Assume that all inclusion scores assigned in a draft tree satisfy Assum. 4.2. Then, the backward sampling stage of Layer Verification (Alg. 2) generates a sequence with identical distribution as the target distribution $q(\cdot)$. (Refer to Appx. F for the proof.)*

Algorithm 1 Forward Node-Scoring Stage of LV

Input: Draft tree $\mathcal{T} = (V, E)$: every $v \in V$ has a draft token X_v and draft (if non-leaf) and target conditional probabilities $p_v(\cdot), q_v(\cdot)$; Single-step verifier "VERIFY"
Output: Inclusion scores & Outgoing flows ($\forall v \in V$)

$\mathsf{a}_0 \leftarrow 1$
$\mathsf{flow}_v(\tilde{x}) \leftarrow 0$ for all $(v, \tilde{x}) \in V \times \Sigma$ s.t. $\mathsf{Ch}[v] = \varnothing$
for $t = 0, 1, \ldots, H-1$ **do**
  $\overline{\mathcal{L}}_t \leftarrow \{v \in \mathcal{L}_t : \mathsf{Ch}[v] \neq \varnothing\}$
  $\lambda_v \leftarrow \mathsf{a}_v / \sum_{v \in \overline{\mathcal{L}}_t} \mathsf{a}_v$
  for each $v \in \overline{\mathcal{L}}_t$ **do in parallel**
    Set source-side cap.'s $p_v(x_{\mathsf{Ch}[v]})$ over $\Sigma^{|\mathsf{Ch}[v]|}$
    Set target-side cap.'s $\frac{\mathsf{a}_v}{\lambda_v} q_v(\tilde{x})$ over Σ
    Set target-side cap. $1 - \frac{\mathsf{a}_v}{\lambda_v}$ of dummy token ($\perp$)
    Get feasible flows $f_v(x_{\mathsf{Ch}[v]} \to \tilde{x})$ using VERIFY
    $\mathsf{flow}_v(\tilde{x}) \leftarrow \lambda_v \cdot \sum_{x_{\mathsf{Ch}[v]}} f_v(x_{\mathsf{Ch}[v]} \to \tilde{x})$ ▷ scaled flow
    for each $w \in \mathsf{Ch}[v]$ **do in parallel**
      $\mathsf{a}_w \leftarrow \frac{\lambda_v \cdot f_v(X_{\mathsf{Ch}[v]} \to X_w)}{p_v(X_{\mathsf{Ch}[v]})} \cdot \frac{1}{\#[X_w \in X_{\mathsf{Ch}[v]}]}$
    end for
  end for
end for

Algorithm 2 Backward Sampling Stage of LV

Input: Draft tree $\mathcal{T} = (V, E)$: every $v \in V$ has a draft token X_v and draft (if non-leaf) and target conditional probabilities $p_v(\cdot), q_v(\cdot)$; $(\mathsf{a}_v, \mathsf{flow}_v(\cdot))_{v \in V}$ from the node scoring stage
Output: Accepted draft path & Corrected token

for $t = H, H-1, \ldots, 0$ **do**
  Compute h_v ($\forall v \in \mathcal{L}_t$) as Eqn. (1)
  $\mathsf{h}_\perp \leftarrow 1 - \sum_{v \in \mathcal{L}_t} \mathsf{h}_v$ ▷ $\perp \notin V$: dummy node
  Sample $v_{\mathsf{out}} \sim \mathsf{Categorical}\big((\mathsf{h}_v)_{v \in \mathcal{L}_t \cup \{\perp\}}\big)$
  if $v_{\mathsf{out}} \in \mathcal{L}_t$ (i.e., $v_{\mathsf{out}} \neq \perp$) **then**
    Sample $y \sim p^{\mathsf{res}}_{v_{\mathsf{out}}}(\cdot)$ (Eqn. (2))
    return $(x_v)_{v \in \mathsf{Path}[v_{\mathsf{out}}]}, y$
  end if ▷ Otherwise, move upward by a layer
end for

4.2. Forward Scoring Stage for Lifting 1-Step Verifier

The backward stage shows that losslessness reduces to a scoring problem: it is enough to construct inclusion scores a_v and outgoing flows $\mathsf{flow}_v(\cdot)$ satisfying Assum. 4.2. The role of the forward stage is precisely to produce such scores and flows. At each nonleaf node v, LV utilizes the chosen single-step verifier to solve a scaled local flow problem so that the resulting outgoing flow satisfies Assum. 4.2. This condition ensures that the residual distribution in Eqn. (2) is valid, which is the key requirement for the backward sampling stage to be lossless.

We start from the root score $\mathsf{a}_0 = 1$ and execute the scoring steps layerwise, moving towards the deepest layer of the draft tree. Suppose that the scores of all nodes in depth t have already been computed. LV then computes the scores for nodes in $\mathcal{L}_{t+1}$: for each non-leaf node $v \in \mathcal{L}_t$, it uses the chosen single-step verifier to send feasible token-level flow from v to its children, under the target-side capacity $\mathsf{a}_v q_v(\cdot)$, following the same principle as in the chain case.

Another main ingredient of LV is a multiplier for draft-side capacities. Let $\overline{\mathcal{L}}_t := \{v \in \mathcal{L}_t : \mathsf{Ch}[v] \neq \varnothing\}$ be the collection of non-leaf nodes in $\mathcal{L}_t$ and $A_t := \sum_{u \in \overline{\mathcal{L}}_t} \mathsf{a}_u$ be the sum of non-leaf nodes in $\mathcal{L}_t$. LV uses the source-side multiplier $\lambda_v := \mathsf{a}_v / A_t$ for each non-leaf node $v \in \overline{\mathcal{L}}_t$. Once these multipliers are determined, LV applies the chosen single-step verifier, or equivalently the corresponding flow-network solver, to the local verification problem at each node v; the draft-side capacities are scaled by λ_v, while the target-side capacities are scaled by a_v. Then, the scores of the children of v are determined by taking the ratio of the child-specific flow to the original, unscaled draft probability of the realized child tuple. Repeating this layerwise recursion from $t = 0$ to $H - 1$ assigns scores to the whole draft tree. Refer to Fig. 3 (Appx. E) for an illustrative example.

Remark. The choice of the base single-step verifier determines how LV routes the local flow at each non-leaf node. In particular, since GR is a near-optimal max-flow solver for single-step multi-candidate verification, LV instantiated with GR as its base verifier inherits this near-optimality when $H = 1$. On the other hand, when the draft is a single chain, the method reduces to the optimal BV rule in the multi-step single-candidate case. To our knowledge, this is the first lossless tree-verification framework that is (near-)optimal for both single-step multi-candidate drafts and multi-step single-candidate drafts.

5. Experiments

Due to the space limit, we defer Tabs. 1, 2 and 3, containing our main experimental results, to Appx. A.4.

5.1. Toy Experiments

Draft Generation. We randomly generate synthetic logits to construct base draft and target distributions, p and q, over a fixed vocabulary set Σ. We generate drafts using three types of draft tree structures: multi-chain, complete-tree, and tapered-tree, each with depth H and maximum branch count (i.e., the maximum number of child nodes) b. Additional details of the synthetic draft-generation procedure are provided in Appx. G.

Compared Methods. We compare LV-lifted single-step verifiers $S \in \{\text{K-SEQ}, \text{RRS}, \text{GR}\}$ to their TV-counterparts: 'LV + S' versus 'TV + S'. We also include two specialized multi-step baselines: TRV and GBV; we evaluate GBV only on multi-chain drafts, as it is specifically designed for them.

Metrics. We report the average number of accepted tokens

per verification call to evaluate and compare the efficiencies of several methods. For statistical significance, we also report standard errors (SE) in parentheses. Also, we measure the total variation distance (TVD, a half of ℓ_1 distance) over Σ^{H+1} between the empirical distribution of generated sequences (by N_{trial} Monte Carlo sampling) and the joint distribution induced by the target q. Every time we sample a draft tree, we call the synthetic verifier once to decode a sequence of length $\leq H+1$, and then sample the remaining tokens from q to fill the generated sequence to length $H+1$. To evaluate losslessness, we also compute the same TVD for samples of the same length drawn directly from q and use this as the baseline: we interpret a verifier as lossless when its TVD matches this baseline.

Results. The main results of our toy experiments are shown in Tab. 1. First, all verification methods match the baseline TVD to three decimal places across all draft structures. The TVD values are not expected to be close to zero, since they are computed between two distributions over a large sequence space. Thus, the relevant comparison is against the direct target sampler, and the observed agreement indicates that LV preserves the target distribution.

We also observe that LV consistently yields longer acceptance lengths than its TV counterpart. Among all evaluated methods, LV-lifted GR achieves the largest accepted length on every draft structure. These results suggest that the benefit of LV is not tied to any particular single-step verifier but rather to the lifting strategy itself. This is especially relevant because several practical speculative-decoding systems, including SpecInfer (Miao et al., 2024), Medusa (Cai et al., 2024), and EAGLE (Li et al., 2024a), can be viewed under our taxonomy as using a TV-lifted RRS. We provide more ablation studies in Appx. G. Across various settings, LV-lifted verifiers consistently accept more tokens than their TV-lifted counterparts while remaining lossless.

5.2. Real-World Benchmark on LLM Decoding

Tasks. We conduct LLM decoding experiments on Spec-Bench (Xia et al., 2024), a standard benchmark for assessing decoding methods of LLMs on diverse domains.

Draft and Target Models. We use Llama3 series models (Grattafiori et al., 2024): Llama3.1-8B-Instruct as the target model and Llama3.2-1B-Instruct as the draft model. Also, we mostly adopt the draft generation and verification pipelines from the PyTorch implementation of EAGLE (Li et al., 2024a). The draft trees are generated from the outputs of several parallel forward passes of the draft model, rather than following the exact EAGLE-based draft construction that uses hidden features as well, since the Llama3-based draft model is not tailored for such a framework.

Compared Methods & Implementation Details. As in the toy experiments, we compare LV and TV, each instantiated with a single-step verifier from $\{\text{K-SEQ}, \text{RRS}, \text{GR}\}$, and (the with-replacement candidate-sampling counterpart of) TRV. Each single-step verifier is converted into a flow network representation and used in the lifting strategy. For a favorable computation, we apply top-k sampling with $k = 20$ for the draft model's candidate token generation across all methods, and we limit the maximum branching count to 2; namely, we test a multi-chain tree (with 2 chains of $H \in \{4, 5\}$) and a complete binary tree ($H \in \{4, 5\}$). Note that the original implementation of RRS is sequential over candidate tokens of each node's children. This is a bit different from our TV + RRS implementation-wise; hence, we also run TV with RRS in its original 'default' form, as in EAGLE's default verification rule.

Results. We report the overall statistics (average & SE) obtained by running 4 random seeds in Tabs. 2 and 3. First, we consistently observe that LV-variants outperform their TV-counterparts in both block efficiency and throughput. In particular, we report $\Delta_{\text{BE}}^{(\text{LV, TV})}$, the percentage improvement in block efficiency of LV over TV, which ranges from 0.5% to 1.8% across settings. Notably, LV-lifted GR achieves the best block efficiency in all settings, while LV-lifted RRS rivals it in decoding speed. We also find that $\Delta_{\text{BE}}^{(\text{LV, TV})}$ escalates as the draft trees deepen, while the speedups compared to the *Baseline*[3] (i.e., decoding without drafts) shrink a bit, possibly due to the need for one more parallel forward pass of 1B-scale draft model per verification round. Moreover, we compute the perplexity of the generated sequences with the target model to confirm generation quality. We also provide detailed results for each of the six different domains of the Spec-Bench dataset in Appx. H.

6. Conclusion

This work identified the lifting strategy for single-step verifiers as an underexplored design axis in speculative tree decoding. While prior multi-step implementations have largely relied on Token Verification (TV), we proposed Layer Verification (LV), a new lifting strategy that coordinates acceptance mass across nodes within the same layer rather than verifying each continuation with purely token-by-token decisions. Theoretically, LV preserves the target autoregressive distribution as long as the single-step verifier it is paired with is lossless. Empirically, toy experiments numerically validate LV's losslessness and show consistent improvements in the average accepted length. On Spec-Bench, LV consistently improves both block efficiency and throughput over TV counterparts across draft structures and single-step verifiers, with LV-lifted GR achieving the strongest block efficiency. Overall, our results reinforce the value of prior advances in single-step verification and highlight lifting as a promising direction for verifier design.

[3] *Baseline*: Block Efficiency = 1 (by definition); Token/s = 35.05 ($\pm$0.01); Perplexity = 1.33.

References

Ankner, Z., Parthasarathy, R., Nrusimha, A., Rinard, C., Ragan-Kelley, J., and Brandon, W. Hydra: Sequentially-dependent draft heads for medusa decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=FbhjirzvJG. 10

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58, 2014. 32

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PEpbUobfJv. 1, 6, 10, 15

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. 10

Chen, J., Liang, Y., and Liu, Z. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026. 10

Chen, Z., May, A., Svirschevski, R., Huang, Y.-H., Ryabinin, M., Jia, Z., and Chen, B. Sequoia: Scalable and robust speculative decoding. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=rk2L9YGDi2. 1, 10, 11

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 32

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022. 12

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, 2024. 10

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 6, 32

Hu, Y., Liu, Z., Dong, Z., Peng, T., McDanel, B., and Zhang, S. Q. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*, 2025a. 1

Hu, Z. and Huang, H. Accelerated speculative sampling based on tree monte carlo. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=stMhi1Sn2G. 11

Hu, Z., Zheng, T., Viswanathan, V., Chen, Z., Rossi, R. A., Wu, Y., Manocha, D., and Huang, H. Towards optimal multi-draft speculative decoding. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025b. URL https://openreview.net/forum?id=9KxnxWOBA5. 11, 15

Jeon, W., Gagrani, M., Goel, R., Park, J., Lee, M., and Lott, C. Recursive speculative decoding: Accelerating LLM inference via sampling without replacement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL https://openreview.net/forum?id=RdKYAHZPxg. 1, 11

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020. 32

Khisti, A. J., Ebrahimi, M., Dbouk, H., Behboodi, A., Memisevic, R., and Louizos, C. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N1L5TgtkAw. 11

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. 32

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 19274–19286. PMLR, 2023. 1, 3, 10, 11, 16

Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. JMLR.org, 2024a. 1, 6, 10, 11, 15, 32

Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b. 1, 10

Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4exx1hUffq. 1, 10

Liu, F., Li, X., Zhao, K., Gao, Y., Zhou, Z., Zhang, Z., Wang, Z., Dou, W., Zhong, S., and Tian, C. Dart: Diffusion-inspired speculative decoding for fast llm inference. *arXiv preprint arXiv:2601.19278*, 2026. 10

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024. 6, 11, 15

Nallapati, R., Zhou, B., Dos Santos, C., Gulçehre, Ç., and Xiang, B. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pp. 280–290, 2016. 32

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018. 1

Sun, R., Zhou, T., Chen, X., and Sun, L. Spechub: Provable acceleration to multi-draft speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20620–20641, 2024. 2, 11, 12, 32

Sun, Z., Suresh, A. T., Ro, J. H., Beirami, A., Jain, H., and Yu, F. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242, 2023. 1, 11

Sun, Z., Mendlovic, U., Leviathan, Y., Aharoni, A., Ro, J. H., Beirami, A., and Suresh, A. T. Block verification accelerates speculative decoding. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=frsg32u0rO. 1, 2, 4, 11, 16

Thomas, R. and Pal, A. Greedy multi-path block verification for faster decoding in speculative sampling. *arXiv preprint arXiv:2602.16961*, 2026a. 2, 11, 16

Thomas, R. K. and Pal, A. Global resolution: Optimal multi-draft speculative sampling via convex optimization. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=gpsczXOsHn. 1, 2, 11, 12, 32

Wang, J., Su, Y., Li, J., Xia, Q., Ye, Z., Duan, X., Wang, Z., and Zhang, M. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. 1, 10

Weng, Y., Hu, Q., Chen, X., Liu, L., Mei, D., Qiu, H., Tian, J., and zhongchao shi. Traversal verification for speculative tree decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=8nOMhDFpkU. 2, 11, 32

Xia, H., Ge, T., Wang, P., Chen, S.-Q., Wei, F., and Sui, Z. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, 2023. 1, 10

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, 2024. 1, 2, 6, 32

Yang, S., Huang, S., Dai, X., and Chen, J. Multi-candidate speculative decoding. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 335–348. Springer, 2025. 1, 11

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao. 32

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF. 10

A. Miscellaneous Addenda

A.1. Broader Impacts

This work studies verification algorithms for speculative decoding and aims to improve the efficiency of autoregressive language model inference. Its potential positive impacts include reducing inference latency, improving hardware utilization, lowering serving costs, and making large language model deployment more accessible. Our work does not introduce new model capabilities, datasets, or deployment mechanisms, and the societal risks are therefore limited. There are no potential risks or negative social impacts that the authors are aware of.

A.2. Limitations

First, our empirical evaluation on Spec-Bench is currently limited to a small number of draft-tree structures, temperatures, and model pairs. We plan to provide additional empirical validation across a wider range of settings in a future revision. Second, Layer Verification requires access to the outgoing flow induced by the underlying single-step verifier to run the backward stage. This information is not necessarily required for Token Verification; therefore, LV may require additional implementation effort for single-step verifiers. In the with-replacement draft-sampling setting considered in our main experiments, the outgoing flow of existing verifiers can be computed efficiently. However, without-replacement sampling over the full vocabulary can make it intractable to compute the outgoing flow exactly. We remark that this issue can be mitigated by restricting draft sampling to the top-k candidate tokens by base-draft probability mass, without harming overall generation quality.

A.3. LLM Usage

We used large language models (LLMs) as auxiliary tools for writing assistance and implementation support. In particular, LLMs were used to polish the paper's writing and grammar and to assist with writing experimental code, especially for the C++/CUDA implementation of the verification algorithms. The core algorithmic ideas, including the design of Layer Verification, the theoretical analysis, and the experimental methodology, were developed by humans and did not originate from LLM-generated suggestions.

A.4. Deferred Results of Main Experiments

Table 1. Toy experiment results, $(|\Sigma|, H, b, \rho, \tau_p, \tau_q, N_{\text{trial}}) = (15, 4, 2, 0.5, 1.0, 1.0, 10^6)$.

Method	Multi-Chain		Tapered-Tree		Complete-Tree	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	2.26 (±0.04)	0.2567	2.50 (±0.04)	0.2568	2.66 (±0.04)	0.2568
LV + K-SEQ	2.51 (±0.03)	0.2568	2.73 (±0.04)	0.2567	2.88 (±0.04)	0.2568
TV + RRS	2.18 (±0.04)	0.2568	2.42 (±0.04)	0.2568	2.47 (±0.04)	0.2567
LV + RRS	2.41 (±0.03)	0.2567	2.61 (±0.04)	0.2567	2.65 (±0.04)	0.2569
TV + GR	2.44 (±0.04)	0.2567	2.70 (±0.05)	0.2568	3.18 (±0.05)	0.2568
LV + GR	2.66 (±0.03)	0.2567	2.88 (±0.04)	0.2568	3.28 (±0.05)	0.2567
TRV	2.45 (±0.03)	0.2568	2.67 (±0.04)	0.2567	2.71 (±0.04)	0.2567
GBV	2.60 (±0.03)	0.2567	–	–	–	–
Baseline	–	0.2569	–	0.2567	–	0.2567

Table 2. LLM decoding experiment on Spec-Bench (with trees of depth 4).

Method	Multi-Chain (4,2)					Complete-Tree (4,2)				
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	PPL	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	PPL
TV + K-SEQ	3.56 (±0.01)		42.38 (±0.13)	+20.9%	1.33	3.70 (±0.00)		43.14 (±0.11)	+23.1%	1.33
LV + K-SEQ	3.60 (±0.01)	+1.12%	42.62 (±0.09)	+21.6%	1.33	3.74 (±0.01)	+1.08%	43.39 (±0.17)	+23.8%	1.33
TV + RRS	3.58 (±0.01)		42.63 (±0.12)	+21.6%	1.33	3.72 (±0.01)		43.75 (±0.18)	+24.8%	1.33
LV + RRS	3.60 (±0.01)	+0.56%	**42.90** (±0.13)	+22.4%	1.33	3.74 (±0.01)	+0.54%	44.09 (±0.15)	+25.8%	1.33
TV + GR	3.59 (±0.01)		42.21 (±0.10)	+20.4%	1.33	3.78 (±0.01)		44.19 (±0.16)	+26.1%	1.33
LV + GR	**3.62** (±0.01)	+0.84%	42.45 (±0.10)	+21.1%	1.33	**3.80** (±0.01)	+0.53%	**44.37** (±0.19)	+26.6%	1.34
default	3.58 (±0.01)		42.30 (±0.09)	+20.7%	1.33	3.72 (±0.01)		43.33 (±0.15)	+23.6%	1.33
TRV	3.61 (±0.00)		42.28 (±0.03)	+20.6%	1.33	3.74 (±0.01)		42.76 (±0.09)	+22.0%	1.33

Table 3. LLM decoding experiment on Spec-Bench (with trees of depth 5).

Method	Multi-Chain (5,2)					Complete-Tree (5,2)				
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	PPL	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	PPL
TV + K-SEQ	3.92 (±0.01)		40.59 (±0.15)	+15.8%	1.33	4.14 (±0.01)		39.73 (±0.07)	+13.4%	1.33
LV + K-SEQ	3.99 (±0.01)	+1.79%	40.90 (±0.05)	+16.7%	1.33	4.20 (±0.02)	+1.45%	39.75 (±0.14)	+13.4%	1.33
TV + RRS	3.94 (±0.01)		40.74 (±0.25)	+16.2%	1.33	4.15 (±0.02)		40.24 (±0.26)	+14.8%	1.33
LV + RRS	3.98 (±0.02)	+1.02%	**41.14** (±0.19)	+17.4%	1.33	4.19 (±0.00)	+0.96%	40.61 (±0.15)	+15.8%	1.33
TV + GR	3.97 (±0.01)		40.70 (±0.20)	+16.1%	1.33	4.23 (±0.02)		40.51 (±0.15)	+15.6%	1.33
LV + GR	**4.03** (±0.02)	+1.51%	40.92 (±0.12)	+16.7%	1.33	**4.29** (±0.01)	+1.42%	**40.71** (±0.15)	+16.1%	1.34
default	3.96 (±0.01)		40.32 (±0.17)	+15.0%	1.33	4.17 (±0.02)		39.84 (±0.18)	+13.7%	1.33
TRV	3.99 (±0.01)		40.47 (±0.04)	+15.4%	1.33	4.19 (±0.01)		38.84 (±0.15)	+10.8%	1.33

B. Related Work

In this section, we briefly review prior research on speculative decoding. We first provide an overview of speculative decoding in Appx. B.1. We then discuss verifier-based methods, covering single-step verifiers in Appx. B.2 and multi-step verifiers in Appx. B.3.

B.1. Speculative Decoding

Speculative decoding accelerates autoregressive generation by employing a lightweight mechanism to draft several future tokens (or a tree of candidate continuations) and a verifier to accept or correct them. Much of the literature studies how to obtain high-quality drafts with low overhead.

One line of work uses an auxiliary drafter. Early methods use a faster auxiliary model, such as a smaller autoregressive approximation model or an independently optimized drafter, to propose multiple tokens before verification by the target model (Xia et al., 2023; Leviathan et al., 2023; Chen et al., 2023). Since the speedup highly relies on draft-target distribution alignment, Zhou et al. (2024) improve the drafter through knowledge distillation. Feature-level drafters such as EAGLE instead exploit hidden states of the target model to improve draft accuracy (Li et al., 2024a;b; 2025).

A second line of work reduces or removes the need for a separate draft language model. Medusa (Cai et al., 2024) augments the target model with multiple lightweight prediction heads; Hydra (Ankner et al., 2024) improves this head-based paradigm by making draft heads sequentially dependent. Self-speculative methods instead reuse the target model itself, for example by exiting at early layers and then verifying or correcting with the remaining layers (Elhoushi et al., 2024).

A third line focuses on the structure and efficiency of the draft itself. SEQUOIA (Chen et al., 2024) and adaptive-tree methods (Wang et al., 2025) optimize draft-tree topology. Recently, parallel and diffusion-inspired drafters such as DART (Liu et al., 2026) and DFlash (Chen et al., 2026) aim to reduce draft-stage latency by predicting multiple future positions in parallel.

B.2. Single-Step Verification Rules

An orthogonal line of work studies the verification problem. We first focus on designing a *single-step* verifier: given one or more drafted next-token candidates, we aim to maximize acceptance while exactly preserving the target distribution. Speculative Sampling (SpS) (Leviathan et al., 2023) solves the single-candidate case via maximal coupling. Sun et al. (2023) formulate the multi-candidate next-token problem as optimal transport and propose an efficient approximation solver named K-SEQ. Jeon et al. (2024); Yang et al. (2025) propose Recursive Rejection Sampling (RRS), which sequentially tests multiple candidates while updating the residual distribution after each rejection. Khisti et al. (2025) introduce an importance-weighted sampler that reduces the multi-candidate problem to a single-candidate setting and design an effective weighted sampler for the special case of two candidates. Most recently, Global Resolution (GR) (Thomas & Pal, 2026b) provides a near-optimal solver for the optimal transport formulation by leveraging polymatroid structure and convex minimization.

Here, K-SEQ, KHISTI, and GR focus on the case where the candidates are sampled in a with-replacement manner, while RRS can handle both with- and without-replacement drafts. Meanwhile, some works (Sun et al., 2024; Hu et al., 2025b) design specialized draft sampling schemes that enable more tractable single-step verifiers.

B.3. Multi-Step Verification Rules

For longer chains or trees, existing approaches fall into two broad umbrellas. The first family extends a single-step verifier via Token Verification (TV): a local next-token rule is applied repeatedly from the root to deeper positions, and verification terminates once the accepted trajectory can no longer be extended. The original chain-based speculative decoding method of Leviathan et al. (2023) follows this pattern. The single-step verifiers discussed above are also evaluated in those works on tree drafts via the same TV-style lifting. In particular, among lossless speculative-decoding methods that use non-greedy probabilistic draft models and whose main contribution lies in draft construction or systems design rather than verification, TV-lifted RRS is the de facto verifier; representative examples include SpecInfer (Miao et al., 2024), EAGLE (Li et al., 2024a), and SEQUOIA (Chen et al., 2024).

The second family instead designs global multi-step verifiers directly, rather than obtaining them through TV. Hu & Huang (2024) introduce Tree Monte Carlo (TMC) and derive a chain verifier that strictly improves upon token verification in the single-chain setting. Sun et al. (2025) propose Block Verification (BV), which jointly verifies an entire drafted block and is optimal for single-chain verification; consequently, on a single chain, BV dominates TV-based lifting of single-step speculative decoding. Weng et al. (2025) extend this line to general trees with Traversal Verification (TRV), a bottom-up, RRS-based procedure that reduces to BV on single-chain drafts and is therefore optimal in that setting. By contrast, in the degenerate single-step case, it collapses to an RRS-style local rule and is thus suboptimal. Finally, Thomas & Pal (2026a) propose Greedy Multi-Path Block Verification (GBV) for the multi-chain setting, where one drafted chain is selected from multiple candidates and then verified using BV under a skewed draft distribution. In summary, TMC and BV are single-chain verifiers, GBV is a multi-chain verifier, and TRV is a tree verifier.

C. Flow Network

In this section, we introduce the basic definitions and concepts of flow networks that are needed to understand the flow-network perspective used in Sec. 3 and 4. For a more detailed introduction, we refer readers to Chapter 26 of Cormen et al. (2022).

A flow network can be viewed as a directed graph through which some resource is routed. The vertices represent intermediate locations, the directed edges represent possible routes, and each edge has a capacity that limits how much resource can be sent along that route. The network has a distinguished source vertex, where flow enters, and a distinguished sink vertex, where flow leaves. A feasible flow must respect the edge capacities and must not create or destroy resources at intermediate vertices.

Formally, let $G = (V, E)$ be a *flow network* with a capacity function $c : V \times V \to \mathbb{R}_{\geq 0}$. We write $c(u \to v)$ for the capacity of the directed edge from u to v, and set $c(u \to v) = 0$ whenever $(u, v) \notin E$. Let s be the *source* of the network, and let t be the *sink*. A *flow* in G is a function $f : V \times V \to \mathbb{R}_{\geq 0}$ satisfying the following two properties:

- **Capacity constraint**: For all $u, v \in V$,

$$0 \leq f(u \to v) \leq c(u \to v).$$

- **Flow-conservation constraint**: For all $u \in V - \{s, t\}$,

$$\sum_{v \in V} f(v \to u) = \sum_{v \in V} f(u \to v).$$

Here, $f(u \to v)$ denotes the amount of flow sent from u to v. The capacity constraint states that the flow on each directed edge must be nonnegative and cannot exceed its capacity. The flow-conservation constraint says that every intermediate vertex simply relays flow: the total amount entering the vertex must equal the total amount leaving it.

The *value* $|f|$ of a flow f is the total amount of flow sent from the source to the sink:

$$|f| = \sum_{v \in V} f(s \to v) - \sum_{v \in V} f(v \to s).$$

In all flow networks considered in this paper, no edge sends flow into the source, so this reduces to

$$|f| = \sum_{v \in V} f(s \to v)$$

The *maximum-flow problem* asks for a feasible flow with the largest possible value. This problem appears in many settings where a limited resource must be routed through constrained intermediate stages, such as water through a pipe network, goods through a supply chain, or materials through a production process. In such examples, the maximum-flow value represents the largest amount of resource that can be transported from the source to the sink without violating any capacity constraint.

Lastly, flow-network perspectives have recently been used to study speculative decoding, particularly for designing and analyzing single-step multi-candidate verifiers. The key observation is that a verification rule can be interpreted as a feasible routing of probability mass in an associated flow network, and maximizing the flow yields an optimal verifier that achieves the maximum acceptance probability (see Sec. 3.1 and Appx. C.1). SpecHub (Sun et al., 2024) formulates the single-step multi-candidate verification problem from optimal transport and linear programming perspectives, and presents an equivalent maximum-flow formulation. Global Resolution (Thomas & Pal, 2026b) further develops this perspective in the same single-step multi-candidate setting by exploiting additional structure of the induced flow network to obtain a more efficient near-optimal solver. Whereas these works use flow networks to analyze single-step verification with multiple next-token candidates, this paper extends the flow-network perspective to the multi-step setting and uses the resulting insights to motivate Layer Verification.

C.1. Omitted Description of Single-Step Multi-Candidate Flow Network

We here allow the draft model to propose k next-token candidates. Then, a verifier can accept a single token among them. Let $X = (X_1, \ldots, X_k)$ be the drafted tuple sampled from a distribution over Σ^k induced by p; let us write the tuple's probability as $p(X)$.

The corresponding flow network is shown in Fig. 2b. The source-to-tuple edge for X has capacity $p(X)$ and the target-to-sink edge for x has capacity $q(x)$. Also, there is an edge from the tuple node X to the token node x if and only if the draft tuple contains x; this reflects that the verifier accepts only a token contained in the sampled tuple. Such a flow f induces a lossless verifier that accepts token $x \in X$ with probability $f(X \to x)/p(X)$, and the sink capacities ensure that the accepted token mass never exceeds q. The resampling distribution for the verifier is defined similarly.

This flow network clarifies why the multi-candidate problem is more difficult. Its first layer has $|\Sigma|^k$ tuple nodes, and the bipartite (sub-)graph between tuples and tokens is dense for moderate k. Thus, the exact maximum flow is generally expensive, and K-SEQ, RRS, and GR can be viewed as efficient feasible-flow constructions; GR is near-optimal under i.i.d. multi-candidate draft sampling.

We leverage the fact that any lossless single-step verification rule can be converted into a flow network with two layers between the source and sink nodes, as illustrated in Fig. 2b. It is a crucial step in the *forward node-scoring stage* of LV, where we use the flow network instantiation to compute scores for each tree node.

D. Detailed Notation & Problem Settings

D.1. Vocabulary and Strings

The vocabulary/alphabet Σ is a finite set of permissible tokens. A string over Σ is a finite sequence of tokens in Σ. We write a concatenation of two strings x and y by '$x \cdot y$', or 'x, y', or 'xy'. We also use a special token EOS to mark the end of a string, indicating where a language model stops the decoding. We denote by $\overline{\Sigma} = \Sigma \cup \{\mathsf{EOS}\}$ an extended vocabulary; we do not explicitly distinguish it from Σ in the main text for the sake of simplicity. Also, the Kleene star '$*$' is used for defining the set of all finite-length strings recursively as

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i, \qquad \text{where } \begin{cases} \Sigma^0 = \{\varepsilon\}, & (\varepsilon : \text{ the empty string}) \\ \Sigma^i = \left\{xy : x \in \Sigma^{i-1}, y \in \Sigma\right\}. & (i \geq 1) \end{cases}$$

We are particularly interested in a language (i.e., a certain subset of strings) $\Sigma^*\{\mathsf{EOS}\} \coloneqq \{x \cdot \mathsf{EOS} : x \in \Sigma^*\} \subset \overline{\Sigma}^*$.

For a string $x \in \overline{\Sigma}^*$, we write $x_{<i} \coloneqq x_1 \cdots x_{i-1}$ and $x_{\leq i} \coloneqq x_1 \cdots x_i$, where $x_{<1} = \varepsilon$. Also, let $x_{i:j} \coloneqq x_i x_{i+1} \cdots x_j$ if $i \leq j$ but $x_{i:j} \coloneqq \varepsilon$ otherwise. More generally, given a sequence of indices J, we often use a shorthand notation $x_J = (x_j)_{j \in J}$.

We denote by $|x|$ the length of a string x, i.e., the number of tokens in x before the leftmost EOS. Namely, while $|\varepsilon| = 0$, if $x \in \overline{\Sigma}^\ell$ for $\ell \geq 1$,

$$|x| = \min\{\ell\} \cup \{i : 1 \leq i \leq \ell - 1, x_{i+1} = \mathsf{EOS}\}.$$

Note that the special token is not counted into the "length". In particular, $|x \cdot \mathsf{EOS}| = |x|$ if $x \in \Sigma^*$. Lastly, we often consider distribution functions defined over $\overline{\Sigma}$ (or $\overline{\Sigma}^*$). Let us assume that every distribution $\mu(\cdot|c)$ (for some $c \in \overline{\Sigma}^*$) of our interests satisfies that $\mu(x|c) = \mathbb{1}[x \in \mathsf{EOS}^*]$ if $\mathsf{EOS} \in c$, i.e., no other tokens than EOS can be generated after EOS from μ.

Draft Tree and Its Construction We consider a tree-shaped draft $\mathcal{T} = (V, E)$ generated by a draft/small model M_s and verified by the target/big model M_b. We index the vertices/nodes of $\mathcal{T}$ as non-negative integers: $V = \{0, \ldots, n\}$. For convenience, we assign node indices in breadth-first search (BFS) order, starting from the root node "0". Given the depth H of $\mathcal{T}$, let us write the layers $\mathcal{L}_t$ of tree $\mathcal{T}$, defined as the collection of all nodes at depth $t = 0, \ldots, H$. Hence, via the BFS numbering of the nodes, it holds that $\mathcal{L}_0 = \{0\}$, $\mathcal{L}_1 = \{1, 2, \ldots\}$, etc.; thereby satisfying

$$0 \leq t < t' \leq H \Longrightarrow \max \mathcal{L}_t + 1 \leq \min \mathcal{L}_{t'}; \quad \text{as well as} \quad \bigcup_{t=0}^{H} \mathcal{L}_t = V.$$

For each node $v \in \{0, \ldots, n\}$, we often use the following notation:

- Depth of v: $t_v = t \iff v \in \mathcal{L}_t$.
- Children, or child nodes, of v: $\mathsf{Ch}[v] \coloneqq \{w \in V : (v, w) \in E\} \subset \mathcal{L}_{t_v+1}$ if $t_v < H$; in particular, $\mathsf{Ch}[v] = \varnothing$ if $t_v = H$.
- Parent of $v \neq 0$: $\mathsf{par}[v] = u \iff u \in \mathsf{Ch}[v]$.
- (Immediate) siblings of v: $\mathsf{Sib}[v] = \mathsf{Ch}[\mathsf{par}[v]]$ if $v \neq 0$; $\mathsf{Sib}[0] = \{0\} = \mathcal{L}_0$.
- The candidate token $x_v \in \overline{\Sigma}$ at $v \geq 1$ (i.e., non-root), generated by the draft model M_s. An exception is that we use $x_0 \in \Sigma^*$ as a given context/query string.
- The path $\mathsf{Path}[v]$ from the root (0) to v, defined as $\mathsf{Path}[v] = (0 = u_0, u_1, \ldots, u_{t_v} = v)$ satisfying $(u_{t-1}, u_t) \in E$ and $u_t \in \mathcal{L}_t$ for all $t \leq t_v$.

Let us write the base draft distribution over $\overline{\Sigma}$ generated by the draft model M_s as p, while we use q for the target distribution, which we equip M_b with. That is to say, the probability that M_s and M_b generate a sequence $x_1 \cdots x_H \in \overline{\Sigma}^H$ given a context $x_0 \in \Sigma^*$ is

$$p(x_1 \cdots x_H \,|\, x_0) = \prod_{i=1}^{H} p(x_i \,|\, x_0 x_{<i}) \;\text{ and }\; q(x_1 \cdots x_H \,|\, x_0) = \prod_{i=1}^{H} q(x_i \,|\, x_0 x_{<i}), \;\text{ respectively.}$$

For the sake of simplicity, let us denote by $\mathbb{P}_v[\cdot]$ the conditional probability for given candidate tokens $(x_i)_{i\in\mathsf{Path}[v]}$ on the path from the root to the node $v \in V$:

$$\mathbb{P}_v[\cdot] \coloneqq \mathbb{P}\left[\cdot \,\middle|\, (x_i)_{i\in\mathsf{Path}[v]}\right].$$

Likewise, for each node $v \in V$, let us define

$$p_v(\cdot) \coloneqq p\left(\cdot \,\middle|\, (x_i)_{i\in\mathsf{Path}[v]}\right); \quad q_v(\cdot) \coloneqq q\left(\cdot \,\middle|\, (x_i)_{i\in\mathsf{Path}[v]}\right).$$

Draft Tree Construction We view draft model M_s as a draft tree generator equipped with a predefined, fixed *tree configuration* (namely, number of nodes, number of children per node, height, etc.), as done in many practical tree-based methods in the literature (Miao et al., 2024; Cai et al., 2024; Li et al., 2024a). Nevertheless, we remark that our theoretical guarantee that LV is lossless remains true whenever the realization/instantiation of children of every draft tree node only depends on the node itself or its ancestors.

There are several strategies to construct a draft tree using the draft distribution p. Here, we review three of them, focusing on how to sample the children's candidate tokens at each node.

- **With-replacement (i.i.d.) sampling.** At each node v, the candidate tokens for its children are sampled independently from p_v. Hence,

$$\mathbb{P}_v\left[(x_j)_{j\in\mathsf{Ch}[v]}\right] = \prod_{j\in\mathsf{Ch}[v]} p_v\left(x_j\right).$$

- **Without-replacement random sampling.** At each node v, the candidate tokens for its children are sampled from p_v without duplicates. It is defined as a sequential sampling process as follows: if we take account of an ordered set of child nodes $\mathsf{Ch}[v] = \{c_v, c_v+1, c_v+2, \cdots\}$,

$$\begin{aligned}
\mathbb{P}_v\left[x_{c_v}\right] &= p_v\left(x_{c_v}\right); \\
\mathbb{P}_v\left[x_{c_v+1} \,\middle|\, x_{c_v}\right] &= \frac{p_v\left(x_{c_v+1}\right)\cdot \mathbb{1}[x_{c_v+1} \neq x_{c_v}]}{1-p_v\left(x_{c_v}\right)}; \\
\mathbb{P}_v\left[x_{c_v+2} \,\middle|\, x_{c_v:(c_v+1)}\right] &= \frac{p_v\left(x_{c_v+2}\right)\cdot \mathbb{1}\left[x_{c_v+2} \notin x_{c_v:(c_v+1)}\right]}{1-\sum_{\ell=c_v}^{c_v+1} p_v\left(x_\ell\right)}; \\
&\vdots \\
\therefore \mathbb{P}_v\left[(x_j)_{j\in\mathsf{Ch}[v]}\right] &= \prod_{j\in\mathsf{Ch}[v]} \frac{p_v\left(x_j\right)\cdot \mathbb{1}\left[x_j \notin x_{c_v:(j-1)}\right]}{1-\sum_{\ell=c_v}^{j-1} p_v\left(x_\ell\right)}.
\end{aligned}$$

- **Almost-greedy sampling.** At each node v with $c = |\mathsf{Ch}[v]| \geq 1$ children, the $c-1$ candidate tokens are deterministically chosen as the top $c-1$ tokens $\sigma_1, \cdots, \sigma_{c-1} \in \overline{\Sigma}$ in terms of p_v, i.e.,

$$p_v\left(\sigma_1\right) \geq p_v\left(\sigma_2\right) \geq \cdots \geq p_v\left(\sigma_{c-1}\right) \geq p_v\left(\sigma\right) \quad (\forall \sigma \notin \sigma_1 \cdots \sigma_{c-1}).$$

  The remaining "random" child is assigned a randomly sampled symbol different from the top $c-1$ symbols. Thus, if we choose $r \in \mathsf{Ch}[v]$ as the index of the random child,

$$\begin{aligned}
&\mathbb{P}_v\left[(x_j)_{j\in\mathsf{Ch}[v]\setminus\{r\}} = \sigma_1\cdots\sigma_{c-1}\right] = 1; \\
&\mathbb{P}_v\left[x_r\right] = \mathbb{P}_v\left[(x_j)_{j\in\mathsf{Ch}[v]}\right] = \frac{p_v\left(x_r\right)\cdot \mathbb{1}[x_r \notin \sigma_1\cdots\sigma_{c-1}]}{1-\sum_{s=1}^{c-1} p_v\left(\sigma_s\right)}.
\end{aligned}$$

  In particular, a specific case with $r_v = \max \mathsf{Ch}[v]$ (placing the random child last) is proposed and analyzed by Hu et al. (2025b), referred to as 'greedy draft sampling'.[4]

Note that, in this paper, we mainly focus on the i.i.d. sampling for simplicity, unless specified otherwise. However, our proposed meta-algorithm can be applied under any of these draft sampling schemes whenever the flow-network formulation of a single-step verifier is computationally tractable.

[4]Do not confuse it with *greedy decoding*, a completely deterministic method with zero temperature.

D.2. Speculative Tree Decoding

The problem setting we are interested in this paper is formally summarized in Alg. 3. It is a strict generalization of the speculative decoding framework studied by Leviathan et al. (2023); Sun et al. (2025): see Alg. 3 of Sun et al. (2025). One of the major features of this framework is that it assumes a class of verification rules (applied in Line 5) that do not modify the probability masses for subsequent verification rounds. Sun et al. (2025) prove the optimality of BV under single-chain drafts in this setting. Notably, Thomas & Pal (2026a) further argue that the optimality holds even outside such a setup. It means that, at least in a multi-step, single-candidate setup, exploring outside this class of verification is not necessary.

Algorithm 3 Speculative Tree Decoding (SPECDEC)

Input:

- Prefix $x \in \Sigma^*$
- Draft tree generator M_s, associated with draft distribution p
- Target model M_b, associated with target distribution q
- Tree verification subroutine: VERIFY (e.g., TV, LV, TRV, GBV)

Output: Decoded string ($\in L = \Sigma^*\{\mathsf{EOS}\}$)

1: **while** $\mathsf{EOS} \notin x$ **do**

2: $x_0 \leftarrow x$

3: Construct $\mathcal{T} = (V, E) \sim M_s(\cdot \mid x_0)$, keeping $p_v(x') = p\left(x' \,\middle|\, x_{\mathsf{Path}[v]}\right)$ for $\forall v \in V, x' \in \overline{\Sigma}$.

4: ▷ Obtain draft tree and conditional probabilities.

5: Call M_b and obtain $q_v(x') = q\left(x' \,\middle|\, x_{\mathsf{Path}[v]}\right)$ for $\forall v \in V, x' \in \overline{\Sigma}$ in parallel. ▷ Parallel scoring.

6: Get the accepted tokens with draft verification: ▷ Draft verification and correction.

$$(X_i)_{i \in \mathsf{Path}[v_{\mathsf{out}}]}, Y = \textsc{Verify}\left(\mathcal{T}, \{p_v(\cdot)\}_{v \in V}, \{q_v(\cdot)\}_{v \in V}\right).$$

7: $x \leftarrow x \cdot (X_i)_{i \in \mathsf{Path}[v_{\mathsf{out}}]} \cdot Y$. ▷ Concatenate decoded tokens to the prefix.

8: **end while**

9: $\ell \leftarrow$ (number of symbols in x), so that $x \in \overline{\Sigma}^{\ell}$

10: $|x| \leftarrow \min\{i : 1 \le i \le \ell - 1, v_{i+1} = \mathsf{EOS}\}$

11: **return** $(x_i)_{i \le |x|} \cdot \mathsf{EOS}$

E. Detailed Implementation of Layer Verification

First, here we provide a more detailed pseudocode for Alg. 2.

Algorithm 4 Backward Sampling Stage of Layer Verification (LV)

Input:

- Draft tree $\mathcal{T} = (V, E)$ of depth $H \geq 1$ with node set $V = \bigcup_{t=0}^{H} \mathcal{L}_t$
- Draft distributions over $\overline{\Sigma}$: $\{p_v(\cdot)\}_{v \in V}$, where $p_v(\cdot) \coloneqq p\left(\cdot \,\middle|\, (x_i)_{i \in \mathsf{Path}[v]}\right)$
- Target distributions over $\overline{\Sigma}$: $\{q_v(\cdot)\}_{v \in V}$, where $q_v(\cdot) \coloneqq q\left(\cdot \,\middle|\, (x_i)_{i \in \mathsf{Path}[v]}\right)$
- Node scoring subroutine: SCORING

Output: Accepted path of candidate tokens $X = (x_v)_{v \in \mathsf{Path}[v_{\mathsf{out}}]} \in \overline{\Sigma}^*$ and corrected token $y \in \overline{\Sigma}$

$(\mathsf{a}_v)_{v \in V}, (\mathsf{flow}_v(\cdot))_{v \in V} \leftarrow \textsc{Scoring}\left(\mathcal{T}, \{p_v(\cdot)\}_{v \in V}, \{q_v(\cdot)\}_{v \in V}\right)$

▷ Node scores (Defn. F.1) and outgoing flows (Eqn. (3)).

for $t = H, H-1, \cdots, 0$ **do**

  $\mathsf{h}_v \leftarrow \frac{\mathsf{a}_v - \sum_{z \in \overline{\Sigma}} \mathsf{flow}_v(z)}{1 - \sum_{v \in \mathcal{L}_t} \sum_{z \in \overline{\Sigma}} \mathsf{flow}_v(z)}$ $(\forall v \in \mathcal{L}_t)$ ▷ Acceptance rates for each node.

  $\mathsf{h}_\perp \leftarrow 1 - \sum_{v \in \mathcal{L}_t} \mathsf{h}_v$ ▷ Rejection rate. (Here, $\perp \notin V$ is a dummy node.)

  Sample $v_{\mathsf{out}} \in \mathcal{L}_t \cup \{\perp\}$ with probability $\mathsf{h}_{v_{\mathsf{out}}}$

  if $v_{\mathsf{out}} \in \mathcal{L}_t$ (equivalently, $v_{\mathsf{out}} \neq \perp$) **then**

    $X \leftarrow (x_v)_{v \in \mathsf{Path}[v_{\mathsf{out}}]}$

    $p^{\mathsf{res}}(\cdot) \leftarrow \mathsf{normalize}\left(q_{v_{\mathsf{out}}}(y) \cdot \mathsf{a}_{v_{\mathsf{out}}} - \mathsf{flow}_{v_{\mathsf{out}}}(y)\right)_{y \in \overline{\Sigma}}$

    ▷ Resampling distribution, where $\mathsf{normalize}\left(\phi(a)\right)_{a \in S} \coloneqq \frac{\phi(\cdot)}{\sum_a \phi(a)}$

    Sample $y \in \overline{\Sigma}$ with probability $p^{\mathsf{res}}(y)$. **return** X, y

  end if

end for

Next, we provide some additional discussion on the "outgoing flow" $\mathsf{flow}_v(z \,|\, \mathsf{draft\_tokens})$: for each node $v \in \mathcal{L}_t$ and a symbol $z \in \overline{\Sigma}$, it is defined as below.

$$\begin{aligned} \mathsf{flow}_v\left(z \,\middle|\, X_{\mathcal{L}_{\leq t}}\right) &\coloneqq \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\leq t+1}}\right) \mathbb{1}[X_w = z] \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \\ &= \sum_{w \in \mathsf{Ch}[v]} \sum_{\substack{x_{\mathcal{L}_{t+1}} \in \overline{\Sigma}^* \\ \text{s.t. } x_w = z}} \mathbb{P}\left[X_{\mathcal{L}_{t+1}} = x_{\mathcal{L}_{t+1}} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \mathsf{a}_w\left(X_{\mathcal{L}_{\leq t}}, x_{\mathcal{L}_{t+1}}\right). \end{aligned} \tag{3}$$

In particular, $\mathsf{flow}_v(z \,|\, \mathsf{draft\_tokens}) = 0$ if $\mathsf{Ch}[v] = \varnothing$.

When we implement the layerwise verification (Alg. 4), we determine each node score $\mathsf{a}_v(\cdot)$ at node v to be independent of draft tokens at nodes not sharing the path from the root to its parent. As a result, the node score satisfies that

$$\mathsf{a}_v\left(x_{\mathcal{L}_{\leq t}}\right) = \mathsf{a}_v\left(\tilde{x}_{\mathcal{L}_{\leq t}}\right) \quad \text{if} \quad \forall u \in \mathsf{Path}[v] \cup \mathsf{Sib}[v] : x_u = \tilde{x}_u,$$

i.e., a_v depends only on the draft token path up to x_v and on the candidate tokens at nodes sharing the same parent. This is a natural assumption in the dependency structure of node scores due to the draft tree sampling, which satisfies the path independence:

$$\mathbb{P}\left[X_{\mathcal{L}_{t+1}} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] = \prod_{v \in \mathcal{L}_t} \mathbb{P}\left[X_{\mathsf{Ch}[v]} \,\middle|\, X_{\mathsf{Path}[v]}\right].$$

To reflect the simplified node-dependency in the node scores, we may write

$$\mathsf{a}_v\left(x_{\mathcal{L}_{\leq t}}\right) = \mathsf{a}_v\left(x_{\mathsf{Path}[v]}, x_{\mathsf{Sib}[v]}\right) = \mathsf{a}_v\left(x_{\mathsf{Path}[u]}, x_{\mathsf{Ch}[u]}\right), \quad \text{if } u = \mathsf{par}[v].$$

Thanks to this, the definition of the outgoing flow in Eqn. (3) can be simplified as

$$\begin{aligned}\mathsf{flow}_v\left(z \,\middle|\, X_{\mathsf{Path}[v]}\right) &= \sum_{w\in\mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathsf{Path}[v]}, X_{\mathsf{Ch}[v]}\right)\mathbb{1}[X_w = z] \,\middle|\, X_{\mathsf{Path}[v]}\right] \\ &= \sum_{w\in\mathsf{Ch}[v]} \sum_{\substack{x_{\mathsf{Ch}[v]}\in\overline{\Sigma}^* \\ \text{s.t. } x_w = z}} \mathbb{P}\left[X_{\mathsf{Ch}[v]} = x_{\mathsf{Ch}[v]} \,\middle|\, X_{\mathsf{Path}[v]}\right] \mathsf{a}_w\left(X_{\mathsf{Path}[v]}, x_{\mathsf{Ch}[v]}\right),\end{aligned}$$

factoring out all the draft-sampling probabilities for $X_{\mathcal{L}_{t+1}\setminus\mathsf{Ch}[v]}$.

Although fully characterizing all node inclusion scores will still yield a valid implementation of LV, relying solely on these scores might be computationally inefficient. In our practical implementation of LV, we compute both the essential node score corresponding to the draft token assigned at the node and the outgoing flows, which provides us with a more computation-efficient implementation.

E.1. Illustration of Layer Verification

Fig. 3 illustrates the forward node-scoring stage of LV.

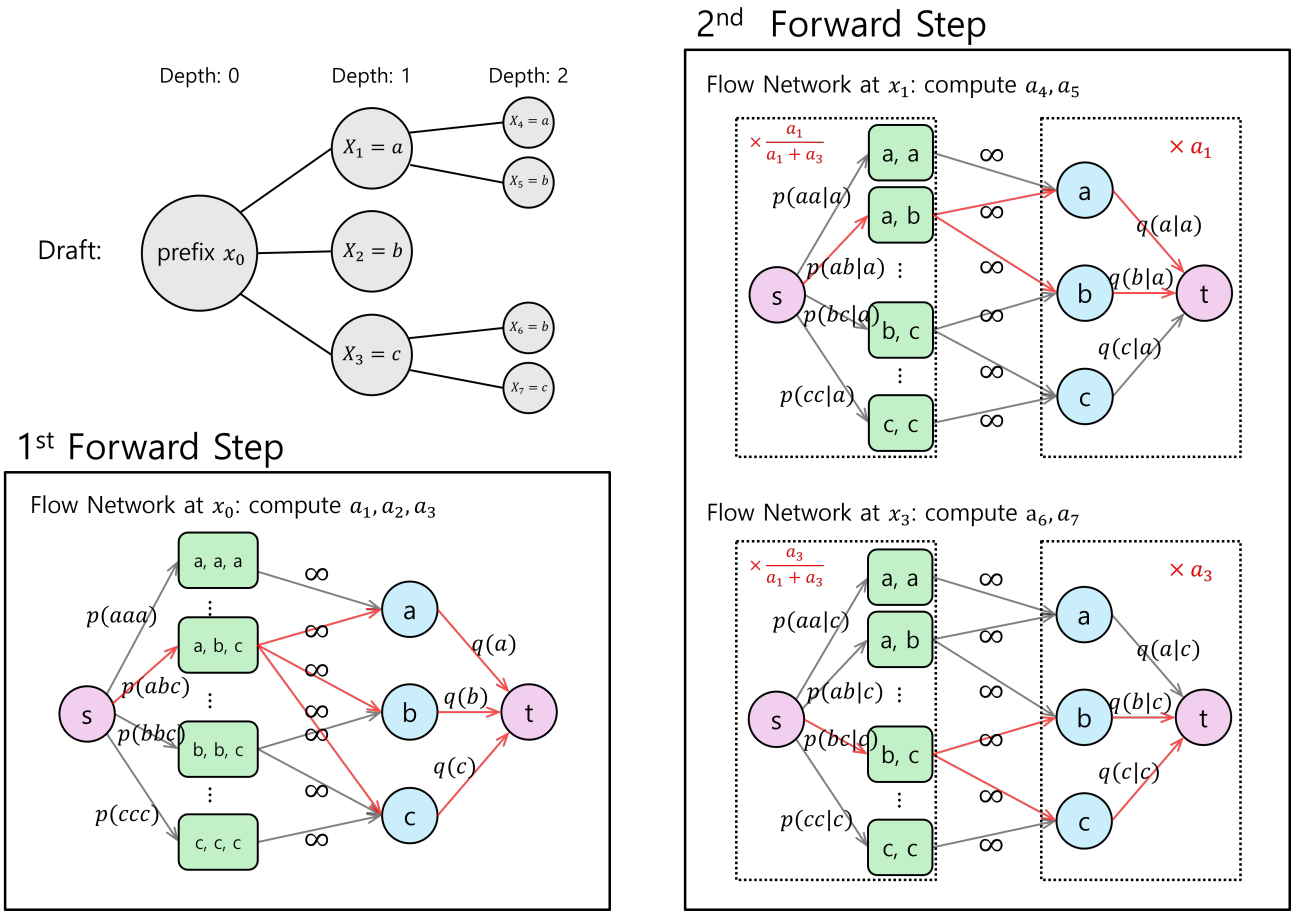


Figure 3. Forward scoring stage of Layer Verification.

In this example, the draft tree has two non-root layers.

The forward node-scoring stage begins by computing a_1, a_2, a_3, which are the scores of the nodes $1, 2, 3 \in \mathcal{L}_1$. This step is exactly the single-step, multi-candidate verification problem described in Appx. C.1. Specifically, the single-step verifier finds a feasible flow f_0 in the flow network shown on the bottom-left side of Fig. 3. For the realized child tuple (a, b, c) at

the root, the scores are assigned by

$$\mathsf{a}_1 = \frac{f_0((a,b,c) \to a)}{p((a,b,c))},$$
$$\mathsf{a}_2 = \frac{f_0((a,b,c) \to b)}{p((a,b,c))},$$
$$\mathsf{a}_3 = \frac{f_0((a,b,c) \to c)}{p((a,b,c))}.$$

We then move to the second step of the forward node-scoring stage. For each nonleaf node in $\mathcal{L}_1$, LV builds a local flow network. The key difference from the root step is that each local network now has two multipliers: λ_v for the draft-side capacities and a_v for the target-side capacities. Here, λ_v is obtained by dividing a_v by the total score of the nonleaf nodes in $\mathcal{L}_1$. In the example, the nonleaf nodes are 1 and 3, so

$$\lambda_1 = \frac{a_1}{a_1 + a_3}, \qquad \lambda_3 = \frac{a_3}{a_1 + a_3}.$$

The right side of Fig. 3 shows these two local flow networks. The single-step verifier finds feasible flows f_1 and f_3 for the networks associated with nodes 1 and 3, respectively. For node 1, whose realized child tuple is (a, b), the scores of its children are assigned by

$$\mathsf{a}_4 = \frac{f_1((a,b) \to a)}{p((a,b) \mid a)},$$
$$\mathsf{a}_5 = \frac{f_1((a,b) \to b)}{p((a,b) \mid a)}.$$

Similarly, for node 3, whose realized child tuple is (b, c), the scores are assigned by

$$\mathsf{a}_6 = \frac{f_3((b,c) \to b)}{p((b,c) \mid c)},$$
$$\mathsf{a}_7 = \frac{f_3((b,c) \to c)}{p((b,c) \mid c)}.$$

Once the multipliers are fixed, the local flow problems for different nonleaf nodes in the same layer are independent. Therefore, the feasible flows for all nodes within one layer can be computed in parallel.

In the general case, the same scoring procedure is applied layer by layer: after all scores in $\mathcal{L}_t$ are computed, LV constructs the scaled local flow problem for every nonleaf node in $\mathcal{L}_t$, solves these local problems in parallel, and uses the resulting flows to assign the scores in $\mathcal{L}_{t+1}$. This continues until scores have been assigned to all nodes in the final layer.

Once the node-scoring phase is complete, LV proceeds to the backward sampling stage. Using the node scores and token-level flows computed in the forward node-scoring phase, this stage completes the algorithm by defining the stopping rule at each node and the corresponding residual distribution. The forward stage is designed so that the mass assigned to accepted draft continuations does not exceed the corresponding target mass. The backward stage then uses the remaining difference between the target mass and the accepted mass to define a valid residual correction step, thereby making LV lossless.

E.2. Using Single-Step Verifiers with Scaled Capacities

Most single-step verifiers are designed for a normalized local verification problem. In the flow-network view, this means that the source-side capacities sum to one and the target-side capacities also sum to one. Let us call this normalized class of local problems $\mathcal{F}$.

In LV, however, the local verification problem at each nonleaf node v may have scaled capacities. Let $p_v(X)$ denote the draft probability of a child tuple x at node v, and let $q_v(z)$ denote the target next-token probability after $\mathsf{Path}[v]$. The local problem that LV needs to solve has source-side capacities sum to λ_v and target-side capacities sum to a_v. Since $\lambda_v = \mathsf{a}_v / A_t$ and $A_t \leq 1$ (note that A_t is the sum of a_u for $u \in \overline{\mathcal{L}}_t$), we have $\mathsf{a}_v \leq \lambda_v$. Let us call this scaled local problem $\mathcal{G}_v$.

The key idea is that any problem in $\mathcal{G}_v$ can be reduced to a normalized problem in $\mathcal{F}$. We divide all capacities in the local problem by λ_v. Then the source-side capacities sum to 1, and the target-side capacities sum to a_v/λ_v. Their total mass is $\mathsf{a}_v/\lambda_v = A_t$, which may be smaller than one.

To make the target side normalized as well, we introduce a dummy token Δ. The dummy token is never proposed by the draft model, so it has zero draft probability. On the target side, it carries the missing mass:

$$\widetilde{q}_v(z) = \frac{\mathsf{a}_v}{\lambda_v} q_v(z), \qquad z \in \Sigma,$$

and

$$\widetilde{q}_v(\Delta) = 1 - \frac{\mathsf{a}_v}{\lambda_v}.$$

Now the source-side capacities sum to one and the target-side capacities also sum to one. Therefore, the transformed problem belongs to the normalized class $\mathcal{F}$, and we can apply the chosen single-step verifier without changing its internal rule.

Let $\widetilde{f}_v$ be the feasible flow returned by the single-step verifier on this transformed problem. We then rescale the real-token part of this flow by λ_v:

$$f_v(X \to z) := \lambda_v \widetilde{f}_v(X \to z), \qquad z \in \Sigma.$$

This gives a feasible flow for the original LV problem. Indeed, the source-side constraint becomes

$$\sum_{z \in \Sigma} f_v(X \to z) \le \lambda_v p_v(X),$$

and the target-side constraint becomes

$$\sum_{X} f_v(X \to z) \le \mathsf{a}_v q_v(z).$$

Thus, the rescaled flow is valid for the scaled local problem $\mathcal{G}_v$.

F. Proofs for Layer Verification

In this section, we provide our rigorous proofs about the validity of our proposed Layer Verification. To this end, we first prove that our choice of *resampling distribution* leads to a lossless draft tree verification rule (Appx. F.1). Subsequently, we provide a derivation of valid *acceptance rates* (Appx. F.2).

Let us summarize necessary notation and problem setting for our proofs.

- $(X_v)_{v \in V}$ be the collection of $\overline{\Sigma}$-valued random variables, each for candidate token of node v. We write x_v as a realization of X_v. One slight exception for the sake of simplicity: we use $X_0 \equiv x_0$ for the given prefix $x_0 \in \Sigma^*$.
- $v_{\mathsf{out}} \in V$ be the node such that the verification algorithm accepts the path $\mathsf{Path}[v_{\mathsf{out}}]$ from the root to the node v_{out}. Also, let $t_{\mathsf{out}} = t_{v_{\mathsf{out}}} \geq 0$ be the length (i.e., the number of non-root nodes) of the accepted path $\mathsf{Path}[v_{\mathsf{out}}]$.
- Y be the corrected token, sampled from a *resampling/residual* distribution conditioned on the candidate tokens in $\mathsf{Path}[v_{\mathsf{out}}]$.
- $Z_{1:t_{\mathsf{out}}} \in \overline{\Sigma}^{t_{\mathsf{out}}}$ be the output of the verification algorithm, i.e., decoded string from a given tree. Also, let us sample $Z_t \sim q\left(\cdot \,|\, Z_{<t}\right)$ for $t_{\mathsf{out}} < t \leq H$, independently from $Z_{1:t_{\mathsf{out}}}$, to complete a *virtually* decoded string $Z_{1:H} \in \overline{\Sigma}^H$. Let z_i be a realization of Z_i. For simplicity, we also use $Z_0 = z_0 = x_0$.
- $\mathcal{L}_{\leq t} = \bigcup_{t=0}^{t} \mathcal{L}_t \subset V$ be the set of all nodes of the draft tree up to depth $t \in \{0, \cdots, H\}$. Similarly, let $\mathcal{L}_{<t} = \bigcup_{t=0}^{t-1} \mathcal{L}_t \subset V$ $(t \in [H])$.
- For notational simplicity, we write $\square_S = (\square_v)_{v \in S}$ for a subset of nodes $S \subset V$.
- We define a partial order between nodes: $v \preceq v'$ iff $\mathsf{Path}[v] \subseteq \mathsf{Path}[v']$.

F.1. LV is a Lossless Tree Verification Rule

We would like to show: for any $t \in [H]$ and $z_{\leq t} \in \overline{\Sigma}^t$,

$$\mathbb{P}\left[Z_{\leq t} = z_{\leq t} \,|\, x_0\right] = q\left(z_{\leq t} \,|\, x_0\right), \tag{Losslessness}$$

i.e., the decoded output from the verification algorithm on a randomly sampled draft tree has the same joint distribution as the target distribution.

To this end, we define the following algorithm-dependent objects properly: the node inclusion scores $\mathsf{a}_v\left(\mathsf{subtree}\right)$ (Defn. F.1) and the resampling distributions $p_v^{\mathsf{res}}\left(\cdot \,|\, \mathsf{subtree}\right)$ (Defn. F.2).

Definition F.1 (Node Inclusion Score for LV; General form of Defn. 4.1)**.** For a node $v \in \mathcal{L}_t$, define its **node inclusion score** as the probability of accepting all nodes in the path from the root to v conditioned on all candidate tokens up to layer t, namely,

$$\mathsf{a}_v\left(X_{\mathcal{L}_{\leq t}}\right) \coloneqq \mathbb{P}\left[v \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right].$$

In particular, $\mathsf{a}_0\left(x_0\right) = 1$, and $\sum_{v \in \mathcal{L}_t} \mathsf{a}_v\left(X_{\mathcal{L}_{\leq t}}\right) = \mathbb{P}\left[t_{\mathsf{out}} \geq t \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \leq 1$.

Definition F.2 (Resampling distribution for LV; See Eqn. (2))**.** For a node $v \in \mathcal{L}_t$, define the **resampling distribution** associated with v, p_v^{res}, as the probability of corrected token Y conditioned on $v_{\mathsf{out}} = v$ (i.e., the verification algorithm accepts the path $\mathsf{Path}[v]$ exactly) given all candidate tokens up to layer t, namely,

$$p_v^{\mathsf{res}}\left(z' \,\middle|\, X_{\mathcal{L}_{\leq t}}\right) \coloneqq \mathbb{P}\left[Y = z' \,\middle|\, X_{\mathcal{L}_{\leq t}},\ v_{\mathsf{out}} = v\right].$$

To prove that LV is lossless, we first assume that the node inclusion scores for all nodes in V are determined, as in the following assumption.

Assumption F.3 (Assum. 4.2)**.** For each depth $t = 0, \ldots, H-1$, node $v \in \mathcal{L}_t$, randomly sampled candidate tokens $X_{\mathcal{L}_{\leq t+1}}$, strings $X_0 = z_0 \in \Sigma^*$, $z_{\leq t} \in \overline{\Sigma}^t$, and token $z' \in \overline{\Sigma}$, the node inclusion scores satisfy the following:

$$q\left(z' \,|\, z_{0:t}\right) \mathsf{a}_v\left(X_{\mathcal{L}_{\leq t}}\right) \geq \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\leq t+1}}\right) \mathbb{1}[X_w = z'] \,\middle|\, X_{\mathcal{L}_{\leq t}}\right],$$

where the expectation is over the randomness of the draft tokens.

Let us choose the resampling distributions based on the given node inclusion scores as follows: for $0 \le t \le H$, $v \in \mathcal{L}_t$, and $X_{\mathcal{L}_{\le t}}$ satisfying $X_{\mathsf{Path}[v]} = x_0 \cdot z_{\le t} = z_{0:t}$,

$$p_v^{\mathsf{res}}\left(z' \,\middle|\, X_{\mathcal{L}_{\le t}}\right) \leftarrow \frac{q\left(z' \,|\, z_{0:t}\right) \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \mathbb{1}[X_w = z'] \,\middle|\, X_{\mathcal{L}_{\le t}}\right]}{\mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right]}, \tag{4}$$

which is identical to the right-hand side of Eqn. (2). In particular, we choose $p_v^{\mathsf{res}}\left(z' \,\middle|\, X_{\mathcal{L}_{\le t}}\right) = q\left(z' \,|\, z_{0:t}\right)$ if v is a leaf node, i.e., $\mathsf{Ch}[v] = \varnothing$ (including the case of $t = H$). We remark that the denominator above is exactly the sum of the numerators across all $z' \in \overline{\Sigma}$. Hence, under Assum. F.3, p_v^{res} is a valid probability function over $\overline{\Sigma}$ for each $v \in V$:

$$p_v^{\mathsf{res}}\left(\cdot \,\middle|\, X_{\mathcal{L}_{\le t_v}}\right) \ge 0 \quad \text{and} \quad \sum_{z' \in \overline{\Sigma}} p_v^{\mathsf{res}}\left(z' \,\middle|\, X_{\mathcal{L}_{\le t_v}}\right) = 1.$$

Proof of Losslessness. Now, we proceed with induction on $t = 1, 2, ..., H$.

Base case: $t = 1$.

$$\begin{aligned}
&\mathbb{P}\left[Z_1 = z_1 \wedge t_{\mathsf{out}} \ge 1 \,|\, x_0\right] \\
&= \mathbb{P}\left[\exists v \in \mathcal{L}_1 : X_v = z_1 \wedge v \preceq v_{\mathsf{out}} \,|\, x_0\right] \\
&= \sum_{v \in \mathcal{L}_1} \mathbb{P}\left[X_v = z_1 \wedge v \preceq v_{\mathsf{out}} \,|\, x_0\right] && \because \text{The events } \{v \preceq v_{\mathsf{out}}\} \text{ are disjoint } (\forall v \in \mathcal{L}_1) \\
&= \sum_{v \in \mathcal{L}_1} \sum_{\substack{x_{\mathcal{L}_1} \in \overline{\Sigma}^* \\ \text{s.t. } x_v = z_1}} \mathbb{P}\left[X_{\mathcal{L}_1} = x_{\mathcal{L}_1} \,|\, x_0\right] \mathbb{P}\left[v \preceq v_{\mathsf{out}} \,|\, x_0, X_{\mathcal{L}_1} = x_{\mathcal{L}_1}\right] \\
&= \sum_{v \in \mathcal{L}_1} \sum_{\substack{x_{\mathcal{L}_1} \in \overline{\Sigma}^* \\ \text{s.t. } x_v = z_1}} \mathbb{P}\left[X_{\mathcal{L}_1} = x_{\mathcal{L}_1} \,|\, x_0\right] \mathsf{a}_v\left(x_{\mathcal{L}_1}\right) && \because \text{By the definition of } \mathsf{a}_v(\cdot) \text{ (Defn. F.1)} \\
&= \sum_{v \in \mathcal{L}_1} \mathbb{E}\left[\mathbb{1}[X_v = z_1] \, \mathsf{a}_v\left(X_{\mathcal{L}_{\le 1}}\right) \,\middle|\, x_0\right].
\end{aligned}$$

Summing this up across $z_1 \in \overline{\Sigma}$ yields that

$$\mathbb{P}\left[t_{\mathsf{out}} \ge 1 \,|\, x_0\right] = \sum_{v \in \mathcal{L}_1} \mathbb{E}\left[\mathsf{a}_v\left(X_{\mathcal{L}_{\le 1}}\right) \,\middle|\, x_0\right].$$

These imply that our particular choice of the resampling distribution (Eqn. (4)) can be rewritten as

$$p_0^{\mathsf{res}}\left(z_1 \,|\, x_0\right) = \frac{q\left(z_1 \,|\, x_0\right) - \mathbb{P}\left[Z_1 = z_1 \wedge t_{\mathsf{out}} \ge 1 \,|\, x_0\right]}{1 - \mathbb{P}\left[t_{\mathsf{out}} \ge 1 \,|\, x_0\right]}.$$

Furthermore, this means that

$$\begin{aligned}
&\mathbb{P}\left[Z_1 = z_1 \wedge t_{\mathsf{out}} = 0 \,|\, x_0\right] \\
&= \mathbb{P}\left[Y = z_1 \,|\, x_0, v_{\mathsf{out}} = 0\right] \cdot \mathbb{P}\left[t_{\mathsf{out}} = 0 \,|\, x_0\right] && \because \text{The only node in layer 0 is the root node ``0''} \\
&= p_0^{\mathsf{res}}\left(z_1 \,|\, x_0\right) \left\{1 - \mathbb{P}\left[t_{\mathsf{out}} \ge 1 \,|\, x_0\right]\right\} && \because \text{By definition of } p_0^{\mathsf{res}} \text{ (Defn. F.2)} \\
&= q\left(z_1 \,|\, x_0\right) - \mathbb{P}\left[Z_1 = z_1, t_{\mathsf{out}} \ge 1 \,|\, x_0\right].
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\mathbb{P}\left[Z_1 = z_1 \,|\, x_0\right] &= \mathbb{P}\left[Z_1 = z_1 \wedge t_{\mathsf{out}} \ge 1 \,|\, x_0\right] + \mathbb{P}\left[Z_1 = z_1 \wedge t_{\mathsf{out}} = 0 \,|\, x_0\right] \\
&= q\left(z_1 \,|\, x_0\right).
\end{aligned}$$

Inductive case: $t \to t+1$. Fix any $t \in [H-1]$ and suppose the following holds:

$$\mathbb{P}\left[Z_{\leq t} = z_{\leq t} \,|\, x_0\right] = q\left(z_{\leq t} \,|\, x_0\right).$$

Observe that the following holds for every $t = 0, \cdots, H-1$.

$$\begin{aligned}
&\mathbb{P}\left[Z_{\leq t+1} = z_{\leq t+1} \wedge t_{\mathsf{out}} \geq t+1 \,|\, x_0\right] \\
&= \mathbb{P}\left[\exists w \in \mathcal{L}_{t+1} : X_{\mathsf{Path}[w]} = z_{0:t+1} \wedge w \preceq v_{\mathsf{out}} \,\middle|\, x_0\right] \\
&= \sum_{w \in \mathcal{L}_{t+1}} \mathbb{P}\left[X_{\mathsf{Path}[w]} = z_{0:t+1} \wedge w \preceq v_{\mathsf{out}} \,\middle|\, x_0\right] && \because \text{The events } \{w \preceq v_{\mathsf{out}}\} \text{ are disjoint } (\forall w \in \mathcal{L}_{t+1}) \\
&= \sum_{w \in \mathcal{L}_{t+1}} \sum_{\substack{x_{\mathcal{L}_{\leq t+1}} \in \overline{\Sigma}^* \\ \text{s.t. } x_{\mathsf{Path}[w]} = z_{0:t+1}}} \mathbb{P}\left[X_{\mathcal{L}_{\leq t+1}} = x_{\mathcal{L}_{\leq t+1}} \,|\, x_0\right] \mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, x_0, X_{\mathcal{L}_{\leq t+1}} = x_{\mathcal{L}_{\leq t+1}}\right] \\
&= \sum_{w \in \mathcal{L}_{t+1}} \sum_{\substack{x_{\mathcal{L}_{\leq t+1}} \in \overline{\Sigma}^* \\ \text{s.t. } x_{\mathsf{Path}[w]} = z_{0:t+1}}} \mathbb{P}\left[X_{\mathcal{L}_{\leq t+1}} = x_{\mathcal{L}_{\leq t+1}} \,|\, x_0\right] \mathsf{a}_w\left(x_{\mathcal{L}_{\leq t+1}}\right) && \because \text{By definition of } \mathsf{a}_v(\cdot) \\
&= \sum_{w \in \mathcal{L}_{t+1}} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[w]} = z_{0:t+1}\right] \mathsf{a}_w\left(X_{\mathcal{L}_{\leq t+1}}\right) \,|\, x_0\right] \\
&= \sum_{v \in \mathcal{L}_t} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[v]} = z_{0:t}\right] \cdot \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\leq t+1}}\right) \mathbb{1}[X_w = z_{t+1}] \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \,\middle|\, x_0\right].
\end{aligned}$$

$$\begin{aligned}
&\mathbb{P}\left[Z_{\leq t+1} = z_{\leq t+1} \wedge t_{\mathsf{out}} = t \,|\, x_0\right] \\
&= \mathbb{P}\left[Y = z_{t+1}, \exists v \in \mathcal{L}_t : X_{\mathsf{Path}[v]} = z_{0:t} \wedge v = v_{\mathsf{out}} \,\middle|\, x_0\right] \\
&= \sum_{v \in \mathcal{L}_t} \mathbb{P}\left[Y = z_{t+1} \wedge X_{\mathsf{Path}[v]} = z_{0:t} \wedge v = v_{\mathsf{out}} \,\middle|\, x_0\right] && \because \text{The events } \{v = v_{\mathsf{out}}\} \text{ are disjoint } (\forall v \in \mathcal{L}_t) \\
&= \sum_{v \in \mathcal{L}_t} \sum_{\substack{x_{\mathcal{L}_{\leq t}} \in \overline{\Sigma}^* \\ \text{s.t. } x_{\mathsf{Path}[v]} = z_{0:t}}} \mathbb{P}\left[X_{\mathcal{L}_{\leq t}} = x_{\mathcal{L}_{\leq t}} \,|\, x_0\right] \mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\leq t}} = x_{\mathcal{L}_{\leq t}}\right] \mathbb{P}\left[Y = z_{t+1} \,\middle|\, X_{\mathcal{L}_{\leq t}} = x_{\mathcal{L}_{\leq t}}, v = v_{\mathsf{out}}\right] \\
&= \sum_{v \in \mathcal{L}_t} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[v]} = z_{0:t}\right] \cdot \mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] \mathbb{P}\left[Y = z_{t+1} \,\middle|\, X_{\mathcal{L}_{\leq t}}, v = v_{\mathsf{out}}\right] \,|\, x_0\right] \\
&= \sum_{v \in \mathcal{L}_t} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[v]} = z_{0:t}\right] \cdot \mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right] p_v^{\mathsf{res}}\left(z_{t+1} \,\middle|\, X_{\mathcal{L}_{\leq t}}\right) \,|\, x_0\right]. && \because \text{By definition of } p_v^{\mathsf{res}}
\end{aligned}$$

$$\begin{aligned}
&\mathbb{P}\left[Z_{\leq t+1} = z_{\leq t+1} \wedge t_{\mathsf{out}} < t \,|\, x_0\right] \\
&= \mathbb{P}\left[Z_{\leq t} = z_{\leq t} \wedge t_{\mathsf{out}} < t \,|\, x_0\right] \mathbb{P}\left[Z_{t+1} = z_{t+1} \,|\, x_0, Z_{\leq t} = z_{\leq t} \wedge t_{\mathsf{out}} < t\right] \\
&= \left\{\mathbb{P}\left[Z_{\leq t} = z_{\leq t} \,|\, x_0\right] - \mathbb{P}\left[Z_{\leq t} = z_{\leq t} \wedge t_{\mathsf{out}} \geq t \,|\, x_0\right]\right\} \cdot q\left(z_{t+1} \,|\, z_{0:t}\right) && \because Z_{t'} \sim q\left(\cdot \,|\, Z_{<t'}\right) \text{ if } t' > t_{\mathsf{out}} \\
&= \left\{q\left(z_{\leq t} \,|\, x_0\right) - \mathbb{P}\left[Z_{\leq t} = z_{\leq t} \wedge t_{\mathsf{out}} \geq t \,|\, x_0\right]\right\} \cdot q\left(z_{t+1} \,|\, z_{0:t}\right) && \because \text{Inductive hypothesis} \\
&= q\left(z_{\leq t+1} \,|\, x_0\right) - \sum_{v \in \mathcal{L}_t} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[v]} = z_{0:t}\right] \, q\left(z_{t+1} \,|\, z_{0:t}\right) \mathsf{a}_v\left(X_{\mathcal{L}_{\leq t}}\right) \,|\, x_0\right]
\end{aligned}$$

Observe that

$$\begin{aligned}\mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] &= \mathbb{P}\left[v \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] - \mathbb{P}\left[v \prec v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] \\ &= \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] \\ &= \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t+1}}\right] \,\middle|\, X_{\mathcal{L}_{\le t}}\right] \\ &= \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right],\end{aligned}$$

which is the same as the denominator of our specific choice of the resampling distribution $p_v^{\mathsf{res}}\left(z_{t+1} \,\middle|\, X_{\mathcal{L}_{\le t}}\right)$ (Eqn. (4)); in other words,

$$\begin{aligned}&\mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] p_v^{\mathsf{res}}\left(z_{t+1} \,\middle|\, X_{\mathcal{L}_{\le t}}\right) \\ &= q\left(z_{t+1} \,|\, z_{0:t}\right) \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \mathbb{1}[X_w = z_{t+1}] \,\middle|\, X_{\mathcal{L}_{\le t}}\right].\end{aligned}$$

Plugging this into the expansion of $\mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} < t \,|\, x_0\right]$, we can derive

$$\begin{aligned}&\mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} < t \,|\, x_0\right] - q\left(z_{\le t+1} \,|\, x_0\right) \\ &= -\sum_{v \in \mathcal{L}_t} \mathbb{E}\left[\mathbb{1}\left[X_{\mathsf{Path}[v]} = z_{0:t}\right]\, q\left(z_{t+1} \,|\, z_{0:t}\right) \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) \,\middle|\, x_0\right] \\ &= -\mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} \ge t+1 \,|\, x_0\right] - \mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} = t \,|\, x_0\right].\end{aligned}$$

Rearranging this, we conclude that

$$\begin{aligned}\mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \,|\, x_0\right] &= \mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} \ge t+1 \,|\, x_0\right] \\ &\quad + \mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} = t \,|\, x_0\right] \\ &\quad + \mathbb{P}\left[Z_{\le t+1} = z_{\le t+1} \wedge t_{\mathsf{out}} < t \,|\, x_0\right] \\ &= q\left(z_{\le t+1} \,|\, x_0\right).\end{aligned}$$

Remark F.4. In fact, we have derived Eqn. (4) in the opposite order as the losslessness proof presented so far. We first have formalized the definition of p_v^{res} as in Defn. F.2. Then, we have reverse-engineered the formula for p_v^{res} which necessarily satisfies (Losslessness) for every $t_v \in [H]$.

F.2. Acceptance Rates of LV

Now, we define and design the acceptance rate at each node of the draft tree.

Definition F.5 (Acceptance rate for LV; See Eqn. (1))**.** For a node $v \in \mathcal{L}_t$, define its **acceptance rate** as the probability of $v_{\mathsf{out}} = v$ given that we know the accepted depth t_{out} is no larger than t (i.e., the verification algorithm rejects everything from depth $> t$) given all candidate tokens up to layer t, namely,

$$\mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right) \coloneqq \mathbb{P}\left[v_{\mathsf{out}} = v \,\middle|\, X_{\mathcal{L}_{\le t}},\ t_{\mathsf{out}} \le t\right].$$

In particular, $\mathsf{h}_0\left(x_0\right) = 1$.

In the validity proof of LV (Appx. F.1), we do not need to define and design the acceptance rules explicitly. However, the *hierarchical structure* of a draft tree introduces a natural recurrence between the node inclusion scores for two consecutive layers. The appropriate design of acceptance rates $\mathsf{h}_v\left(\cdot\right)$ for our algorithm appears spontaneously from the recurrence.

With this sketch of derivation, we *aim* to derive the acceptance rates as: for $0 \le t \le H$ and $v \in \mathcal{L}_t$,

$$\mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right) \leftarrow \frac{\mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) - \sum_{w \in \mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right]}{1 - \sum_{w \in \mathcal{L}_{t+1}} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right]}, \tag{5}$$

which is identical to the Eqn. (1). In particular, if $t = H$ and $v \in \mathcal{L}_H$, then $\mathsf{h}_v\left(X_{\mathcal{L}_{\le H}}\right) = \mathsf{a}_v\left(X_{\mathcal{L}_{\le H}}\right)$ ($\because \mathsf{Ch}[v] = \mathcal{L}_{H+1} = \varnothing$). We also remark that Eqn. (5) satisfies $\mathsf{h}_v\left(X_{\mathcal{L}_{\le t+1}}\right) \in [0,1]$, thereby having a valid range of values if Assum. F.3 holds. This is because $\bigcup_{v\in\mathcal{L}_t} \mathsf{Ch}[v] = \mathcal{L}_{t+1}$ and $\sum_{v\in\mathcal{L}_t} \mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) \le 1$ (see Defn. F.1).

Derivation of Eqn. (5). If $t = H$ and $v \in \mathcal{L}_H$, then by Defn. F.5,

$$\mathsf{h}_v\left(X_{\mathcal{L}_{\le H}}\right) = \mathbb{P}\left[v_{\mathsf{out}} = v \,\middle|\, X_{\mathcal{L}_{\le H}},\ t_{\mathsf{out}} \le H\right] = \mathbb{P}\left[v_{\mathsf{out}} = v \,\middle|\, X_{\mathcal{L}_{\le H}}\right] = \mathsf{a}_v\left(x_{\mathcal{L}_{\le H}}\right).$$

Thus, let us fix any $t \in \{0,\dots,H-1\}$ and $v \in \mathcal{L}_t$. We begin with the decomposition of $\mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right)$.

$$\begin{aligned}\mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right) &= \mathbb{P}\left[v \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] && \because \text{Defn. F.1}\\ &= \mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] + \mathbb{P}\left[v \prec v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right].\end{aligned}$$

The first term can be manipulated as

$$\begin{aligned}&\mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\\ &= \mathbb{P}\left[t_{\mathsf{out}} \le t \wedge v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\\ &= \mathbb{P}\left[t_{\mathsf{out}} \le t \,\middle|\, X_{\mathcal{L}_{\le t}}\right] \cdot \mathbb{P}\left[v = v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}, t_{\mathsf{out}} \le t\right]\\ &= \left(1 - \mathbb{P}\left[t_{\mathsf{out}} \ge t+1 \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\right) \cdot \mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right) && \because \text{Defn. F.5}\\ &= \left(1 - \mathbb{E}\left[\mathbb{P}\left[t_{\mathsf{out}} \ge t+1 \,\middle|\, X_{\mathcal{L}_{\le t+1}}\right] \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\right) \cdot \mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right) && \because \text{Tower law}\\ &= \left(1 - \mathbb{E}\left[\sum_{w\in\mathcal{L}_{t+1}} \mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t+1}}\right] \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\right) \cdot \mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right)\\ &= \left(1 - \sum_{w\in\mathcal{L}_{t+1}} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\right) \cdot \mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right).\end{aligned}$$

On the other hand, the second term is simply

$$\begin{aligned}\mathbb{P}\left[v \prec v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right] &= \sum_{w\in\mathsf{Ch}[v]} \mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\\ &= \sum_{w\in\mathsf{Ch}[v]} \mathbb{E}\left[\mathbb{P}\left[w \preceq v_{\mathsf{out}} \,\middle|\, X_{\mathcal{L}_{\le t+1}}\right] \,\middle|\, X_{\mathcal{L}_{\le t}}\right] && \because \text{Tower law}\\ &= \sum_{w\in\mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right].\end{aligned}$$

Combining these, we have a recurrence between inclusion scores of v and $\forall w \in \mathcal{L}_{t+1}$:

$$\begin{aligned}&\mathsf{a}_v\left(X_{\mathcal{L}_{\le t}}\right)\\ &= \left(1 - \sum_{w\in\mathcal{L}_{t+1}} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right]\right) \cdot \mathsf{h}_v\left(X_{\mathcal{L}_{\le t}}\right) + \sum_{w\in\mathsf{Ch}[v]} \mathbb{E}\left[\mathsf{a}_w\left(X_{\mathcal{L}_{\le t+1}}\right) \,\middle|\, X_{\mathcal{L}_{\le t}}\right].\end{aligned}$$

Finally, we obtain Eqn. (5) by rearranging the terms in the equation above.

G. Additional Details of Toy Experiments

The synthetic draft and target distributions are controlled by three parameters: a similarity parameter ρ and two temperature parameters τ_p and τ_q. Concretely, we draw a shared Gaussian component u_c and two independent Gaussian perturbations $\epsilon_c^p, \epsilon_c^q$. For a chosen $0 \le \rho \le 1$, the draft and target logits are defined as

$$\ell_c^p = \rho u_c + (1-\rho)\epsilon_c^p, \qquad \ell_c^q = \rho u_c + (1-\rho)\epsilon_c^q.$$

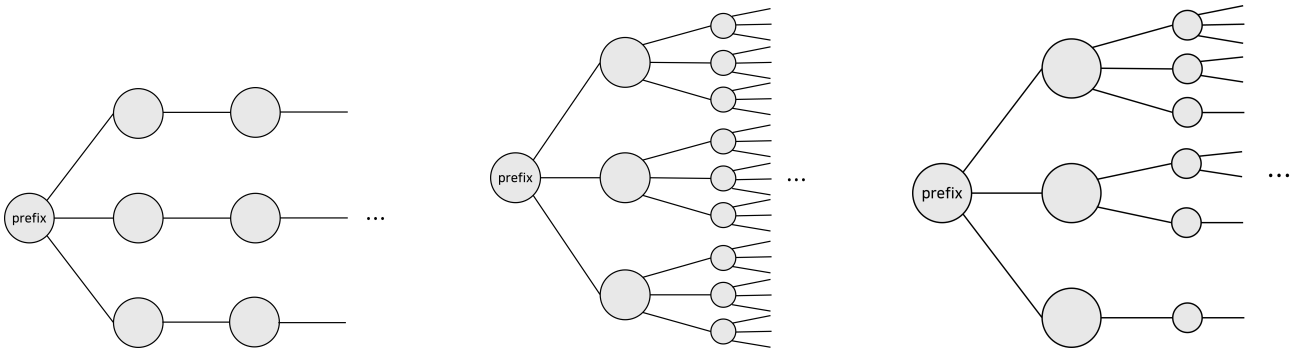
The conditional draft and target distributions are then obtained by applying softmax with temperatures τ_p and τ_q, respectively:

$$p(\cdot|c) = \mathrm{softmax}(\ell_c^p/\tau_p), \qquad q(\cdot|c) = \mathrm{softmax}(\ell_c^q/\tau_q).$$

Our toy experiments involve 3 tree-draft structures: multi-chain, complete-tree, and tapered-tree, each parameterized by a branching budget b. In a multi-chain draft, only the root branches with branching budget b, and each child then continues as a chain. In a complete-tree draft, every internal node has exactly b children. The tapered-tree topology is designed to lie between the other two extremes. The root is first expanded into b children. For any non-root node v, let idx_v be the sibling index of v (starting from zero) among those children. Then, the number of children assigned to v is

$$|\mathsf{Ch}[v]| = \max\left\{|\mathsf{Sib}[v]| - \mathrm{idx}_v,\ 1\right\},$$

where $\mathsf{par}[v]$ denotes its parent and $|\mathsf{Sib}[v]|$ is the number of children of its parent (i.e., immediate sibling nodes). The example draft structures for $b = 3$ are illustrated in Fig. 4.



(a) Multi-chain *(b)* Complete-tree *(c)* Tapered-tree

Figure 4. Draft structures with $b = 3$.

Computation detail of TVDs. To evaluate losslessness, we complete every verifier output to length $H+1$ using exact target sampling. Specifically, if a verifier accepts a prefix $x_{1:a}$ and then samples a corrected token y, we form $z_{1:a+1} = (x_{1:a}, y)$, and then sample $z_{a+2:H+1}$ autoregressively from the target distribution q. Given N Monte Carlo samples, we estimate the empirical distribution $\widehat{q}_{\mathrm{dec}}$ of completed sequences and compute the total variation distance:

$$\mathrm{TVD}(\widehat{q}_{\mathrm{dec}}, q) = \frac{1}{2} \sum_{z \in \Sigma^{H+1}} \left|\widehat{q}_{\mathrm{dec}}(z) - q(z)\right|.$$

In what follows, we report the ablation results. Unless otherwise stated, we use the baseline configuration

$$(|\Sigma|, H, b, \rho, \tau_p, \tau_q, N_{\mathrm{trial}}) = (15, 4, 2, 0.5, 1.0, 1.0, 10^6).$$

We ablate one factor at a time while holding all other factors at their baseline values. Specifically, we consider the following:

1. Depth: $H \in \{2, 3, 4, 5\} \to$ Tabs. 4, 5 and 6;
2. Branching budget: $b \in \{1, 2, 3\} \to$ Tabs. 7, 8 and 9;
3. Temperature: $(\tau_p, \tau_q) \in \{(0.5, 0.5), (0.75, 0.75), (1.0, 1.0), (1.5, 1.5)\} \to$ Tabs. 13, 14 and 15;
4. Similarity: $\rho \in \{0.0, 0.25, 0.5\} \to$ Tabs. 10, 11 and 12.

We report the metrics with mean $\pm$ standard errors (SEs) over 20 random seeds. These ablations serve two purposes: (1) to verify that losslessness is maintained, and (2) to evaluate whether each LV-lifted verifier achieves a higher average accepted length than its TV-lifted counterpart.

All experiments were run for 30 hours on a system with two AMD EPYC 7763 64-core processors.

Table 4. Toy experiment ablation on tree depth H with Complete-Tree drafts.

Method	$H=2$		$H=3$		$H=4$		$H=5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.55 (±0.02)	0.0622	2.15 (±0.03)	0.2257	2.66 (±0.04)	0.6219	3.09 (±0.05)	0.9177
LV + K-SEQ	1.58 (±0.02)	0.0621	2.25 (±0.03)	0.2258	2.88 (±0.04)	0.6219	3.47 (±0.05)	0.9176
TV + RRS	1.48 (±0.02)	0.0619	2.02 (±0.03)	0.2254	2.47 (±0.04)	0.6219	2.83 (±0.05)	0.9176
LV + RRS	1.51 (±0.02)	0.0619	2.11 (±0.03)	0.2257	2.65 (±0.04)	0.6221	3.13 (±0.04)	0.9176
TV + GR	1.73 (±0.03)	0.0620	2.49 (±0.04)	0.2257	3.18 (±0.05)	0.6219	3.82 (±0.06)	0.9176
LV + GR	1.75 (±0.03)	0.0624	2.54 (±0.04)	0.2255	3.29 (±0.05)	0.6218	4.00 (±0.06)	0.9175
TRV	1.52 (±0.02)	0.0620	2.15 (±0.03)	0.2259	2.71 (±0.04)	0.6219	3.23 (±0.05)	0.9176
Baseline	–	0.0619	–	0.2257	–	0.6217	–	0.9174

Table 5. Toy experiment ablation on tree depth H with Tapered-Tree drafts.

Method	$H=2$		$H=3$		$H=4$		$H=5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.52 (±0.03)	0.0621	2.07 (±0.04)	0.2256	2.50 (±0.04)	0.6219	2.83 (±0.05)	0.9176
LV + K-SEQ	1.56 (±0.02)	0.0616	2.19 (±0.03)	0.2260	2.73 (±0.04)	0.6219	3.22 (±0.04)	0.9176
TV + RRS	1.47 (±0.02)	0.0619	2.00 (±0.03)	0.2254	2.42 (±0.04)	0.6219	2.76 (±0.05)	0.9175
LV + RRS	1.50 (±0.02)	0.0619	2.09 (±0.03)	0.2257	2.61 (±0.04)	0.6219	3.07 (±0.04)	0.9175
TV + GR	1.66 (±0.03)	0.0616	2.25 (±0.04)	0.2256	2.70 (±0.05)	0.6218	3.04 (±0.05)	0.9175
LV + GR	1.68 (±0.03)	0.0623	2.32 (±0.03)	0.2256	2.88 (±0.04)	0.6219	3.36 (±0.04)	0.9177
TRV	1.51 (±0.02)	0.0619	2.13 (±0.03)	0.2255	2.67 (±0.04)	0.6216	3.16 (±0.05)	0.9175
Baseline	–	0.0621	–	0.2259	–	0.6216	–	0.9175

Table 6. Toy experiment ablation on tree depth H with Multi-Chain drafts.

Method	$H=2$		$H=3$		$H=4$		$H=5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.46 (±0.02)	0.0620	1.92 (±0.03)	0.2257	2.26 (±0.04)	0.6219	2.51 (±0.04)	0.9175
LV + K-SEQ	1.50 (±0.02)	0.0620	2.05 (±0.03)	0.2260	2.51 (±0.03)	0.6221	2.92 (±0.04)	0.9176
TV + RRS	1.41 (±0.02)	0.0620	1.85 (±0.03)	0.2255	2.18 (±0.04)	0.6221	2.42 (±0.04)	0.9176
LV + RRS	1.44 (±0.02)	0.0621	1.97 (±0.03)	0.2260	2.41 (±0.03)	0.6223	2.79 (±0.04)	0.9176
TV + GR	1.58 (±0.03)	0.0621	2.07 (±0.04)	0.2259	2.44 (±0.04)	0.6219	2.71 (±0.05)	0.9176
LV + GR	1.60 (±0.02)	0.0621	2.17 (±0.03)	0.2255	2.66 (±0.03)	0.6216	3.08 (±0.04)	0.9177
TRV	1.46 (±0.02)	0.0621	1.99 (±0.03)	0.2253	2.45 (±0.03)	0.6223	2.84 (±0.04)	0.9175
GBV	1.56 (±0.02)	0.0623	2.13 (±0.03)	0.2257	2.60 (±0.03)	0.6220	3.00 (±0.04)	0.9177
Baseline	–	0.0619	–	0.2255	–	0.6217	–	0.9177

Table 7. Toy experiment ablation on branching counts with Complete-Tree drafts.

Method	$b = 1$		$b = 2$		$b = 3$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.96 ($\pm$0.04)	0.6219	2.66 ($\pm$0.04)	0.6219	3.04 ($\pm$0.04)	0.6217
LV + K-SEQ	2.21 ($\pm$0.04)	0.6219	2.88 ($\pm$0.04)	0.6219	3.21 ($\pm$0.04)	0.6219
TV + RRS	1.97 ($\pm$0.04)	0.6221	2.47 ($\pm$0.04)	0.6219	2.75 ($\pm$0.04)	0.6219
LV + RRS	2.22 ($\pm$0.04)	0.6218	2.65 ($\pm$0.04)	0.6221	2.89 ($\pm$0.04)	0.6218
TV + GR	1.97 ($\pm$0.04)	0.6218	3.18 ($\pm$0.05)	0.6219	3.70 ($\pm$0.04)	0.6219
LV + GR	2.21 ($\pm$0.04)	0.6220	3.29 ($\pm$0.05)	0.6218	3.73 ($\pm$0.04)	0.6219
TRV	2.22 ($\pm$0.04)	0.6218	2.71 ($\pm$0.04)	0.6219	2.97 ($\pm$0.04)	0.6218
Baseline	–	0.6219	–	0.6217	–	0.6219

Table 8. Toy experiment ablation on branching counts with Tapered-Tree drafts.

Method	$b = 1$		$b = 2$		$b = 3$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.96 ($\pm$0.04)	0.6219	2.50 ($\pm$0.04)	0.6219	2.82 ($\pm$0.04)	0.6218
LV + K-SEQ	2.21 ($\pm$0.04)	0.6219	2.73 ($\pm$0.04)	0.6219	3.02 ($\pm$0.04)	0.6218
TV + RRS	1.97 ($\pm$0.04)	0.6221	2.42 ($\pm$0.04)	0.6219	2.69 ($\pm$0.04)	0.6220
LV + RRS	2.22 ($\pm$0.04)	0.6218	2.61 ($\pm$0.04)	0.6219	2.84 ($\pm$0.04)	0.6220
TV + GR	1.97 ($\pm$0.04)	0.6218	2.70 ($\pm$0.05)	0.6218	3.06 ($\pm$0.03)	0.6218
LV + GR	2.21 ($\pm$0.04)	0.6220	2.88 ($\pm$0.04)	0.6219	3.18 ($\pm$0.03)	0.6219
TRV	2.22 ($\pm$0.04)	0.6218	2.67 ($\pm$0.04)	0.6216	2.93 ($\pm$0.04)	0.6219
Baseline	–	0.6219	–	0.6216	–	0.6218

Table 9. Toy experiment ablation on branching counts with Multi-Chain drafts.

Method	$b = 1$		$b = 2$		$b = 3$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.96 ($\pm$0.04)	0.6219	2.26 ($\pm$0.04)	0.6219	2.39 ($\pm$0.03)	0.6220
LV + K-SEQ	2.21 ($\pm$0.04)	0.6219	2.51 ($\pm$0.03)	0.6221	2.64 ($\pm$0.03)	0.6220
TV + RRS	1.97 ($\pm$0.04)	0.6221	2.18 ($\pm$0.04)	0.6221	2.29 ($\pm$0.03)	0.6215
LV + RRS	2.22 ($\pm$0.04)	0.6218	2.41 ($\pm$0.03)	0.6223	2.51 ($\pm$0.03)	0.6218
TV + GR	1.97 ($\pm$0.04)	0.6218	2.44 ($\pm$0.04)	0.6219	2.61 ($\pm$0.03)	0.6218
LV + GR	2.21 ($\pm$0.04)	0.6220	2.66 ($\pm$0.03)	0.6216	2.80 ($\pm$0.02)	0.6220
TRV	2.22 ($\pm$0.04)	0.6218	2.45 ($\pm$0.03)	0.6223	2.57 ($\pm$0.03)	0.6219
GBV	2.22 ($\pm$0.04)	0.6220	2.60 ($\pm$0.03)	0.6220	2.55 ($\pm$0.03)	0.6221
Baseline	–	0.6218	–	0.6217	–	0.6221

Table 10. Toy experiment ablation on distribution similarity (ρ) with Complete-Tree drafts.

Method	$\rho = 0.0$		$\rho = 0.25$		$\rho = 0.5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.42 (±0.07)	0.4608	1.95 (±0.06)	0.5721	2.66 (±0.04)	0.6219
LV + K-SEQ	1.66 (±0.07)	0.4612	2.21 (±0.06)	0.5723	2.88 (±0.04)	0.6219
TV + RRS	1.34 (±0.06)	0.4607	1.82 (±0.06)	0.5722	2.47 (±0.04)	0.6219
LV + RRS	1.51 (±0.06)	0.4605	2.02 (±0.05)	0.5724	2.65 (±0.04)	0.6221
TV + GR	1.58 (±0.08)	0.4605	2.26 (±0.08)	0.5723	3.18 (±0.05)	0.6219
LV + GR	1.80 (±0.08)	0.4608	2.46 (±0.07)	0.5720	3.29 (±0.05)	0.6218
TRV	1.54 (±0.07)	0.4604	2.07 (±0.06)	0.5723	2.71 (±0.04)	0.6219
Baseline	–	0.4608	–	0.5721	–	0.6217

Table 11. Toy experiment ablation on distribution similarity (ρ) with Tapered-Tree drafts.

Method	$\rho = 0.0$		$\rho = 0.25$		$\rho = 0.5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.31 (±0.06)	0.4608	1.80 (±0.06)	0.5718	2.50 (±0.04)	0.6219
LV + K-SEQ	1.52 (±0.07)	0.4606	2.06 (±0.06)	0.5724	2.73 (±0.04)	0.6219
TV + RRS	1.30 (±0.06)	0.4604	1.78 (±0.05)	0.5720	2.42 (±0.04)	0.6219
LV + RRS	1.47 (±0.06)	0.4608	1.97 (±0.05)	0.5722	2.61 (±0.04)	0.6219
TV + GR	1.37 (±0.07)	0.4608	1.93 (±0.06)	0.5723	2.70 (±0.05)	0.6218
LV + GR	1.58 (±0.07)	0.4606	2.15 (±0.06)	0.5719	2.88 (±0.04)	0.6219
TRV	1.50 (±0.07)	0.4607	2.02 (±0.06)	0.5726	2.67 (±0.04)	0.6216
Baseline	–	0.4606	–	0.5722	–	0.6216

Table 12. Toy experiment ablation on distribution similarity (ρ) with Multi-Chain drafts.

Method	$\rho = 0.0$		$\rho = 0.25$		$\rho = 0.5$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.18 (±0.06)	0.4608	1.63 (±0.05)	0.5720	2.26 (±0.04)	0.6219
LV + K-SEQ	1.39 (±0.06)	0.4607	1.88 (±0.05)	0.5722	2.51 (±0.03)	0.6221
TV + RRS	1.15 (±0.05)	0.4604	1.58 (±0.05)	0.5724	2.18 (±0.04)	0.6221
LV + RRS	1.32 (±0.06)	0.4610	1.79 (±0.05)	0.5723	2.41 (±0.03)	0.6223
TV + GR	1.25 (±0.06)	0.4609	1.74 (±0.06)	0.5719	2.44 (±0.04)	0.6219
LV + GR	1.46 (±0.06)	0.4606	1.98 (±0.05)	0.5721	2.66 (±0.03)	0.6216
TRV	1.34 (±0.06)	0.4609	1.82 (±0.05)	0.5722	2.45 (±0.03)	0.6223
GBV	1.47 (±0.06)	0.4606	1.97 (±0.05)	0.5722	2.60 (±0.03)	0.6220
Baseline	–	0.4607	–	0.5719	–	0.6217

Table 13. Toy experiment ablation on temperature with Complete-Tree drafts.

Method	$(\tau_p, \tau_q) = (0.5, 0.5)$		$(\tau_p, \tau_q) = (0.75, 0.75)$		$(\tau_p, \tau_q) = (1.0, 1.0)$		$(\tau_p, \tau_q) = (1.5, 1.5)$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.55 (±0.08)	0.2919	2.20 (±0.06)	0.4934	2.66 (±0.04)	0.6219	3.17 (±0.03)	0.7511
LV + K-SEQ	1.77 (±0.08)	0.2919	2.45 (±0.06)	0.4936	2.88 (±0.04)	0.6219	3.33 (±0.02)	0.7512
TV + RRS	1.47 (±0.08)	0.2915	2.06 (±0.06)	0.4935	2.47 (±0.04)	0.6219	2.95 (±0.03)	0.7514
LV + RRS	1.64 (±0.08)	0.2918	2.25 (±0.05)	0.4939	2.65 (±0.04)	0.6221	3.10 (±0.02)	0.7515
TV + GR	1.72 (±0.10)	0.2915	2.58 (±0.07)	0.4937	3.18 (±0.05)	0.6219	3.79 (±0.02)	0.7512
LV + GR	1.93 (±0.09)	0.2917	2.75 (±0.07)	0.4935	3.29 (±0.05)	0.6218	3.80 (±0.02)	0.7511
TRV	1.67 (±0.08)	0.2916	2.31 (±0.05)	0.4935	2.71 (±0.04)	0.6219	3.15 (±0.02)	0.7513
Baseline	–	0.2918	–	0.4932	–	0.6217	–	0.7511

Table 14. Toy experiment ablation on temperature with Tapered-Tree drafts.

Method	$(\tau_p, \tau_q) = (0.5, 0.5)$		$(\tau_p, \tau_q) = (0.75, 0.75)$		$(\tau_p, \tau_q) = (1.0, 1.0)$		$(\tau_p, \tau_q) = (1.5, 1.5)$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.43 (±0.08)	0.2916	2.05 (±0.06)	0.4939	2.50 (±0.04)	0.6219	3.04 (±0.03)	0.7511
LV + K-SEQ	1.64 (±0.08)	0.2916	2.30 (±0.05)	0.4936	2.73 (±0.04)	0.6219	3.21 (±0.02)	0.7514
TV + RRS	1.42 (±0.07)	0.2918	2.00 (±0.05)	0.4936	2.42 (±0.04)	0.6219	2.92 (±0.03)	0.7511
LV + RRS	1.59 (±0.07)	0.2912	2.21 (±0.05)	0.4938	2.61 (±0.04)	0.6219	3.07 (±0.02)	0.7513
TV + GR	1.50 (±0.08)	0.2916	2.19 (±0.06)	0.4938	2.70 (±0.05)	0.6218	3.26 (±0.02)	0.7510
LV + GR	1.70 (±0.08)	0.2918	2.41 (±0.06)	0.4938	2.88 (±0.04)	0.6219	3.36 (±0.02)	0.7512
TRV	1.62 (±0.08)	0.2915	2.26 (±0.05)	0.4937	2.67 (±0.04)	0.6216	3.13 (±0.02)	0.7512
Baseline	–	0.2913	–	0.4934	–	0.6216	–	0.7508

Table 15. Toy experiment ablation on temperature with Multi-Chain drafts.

Method	$(\tau_p, \tau_q) = (0.5, 0.5)$		$(\tau_p, \tau_q) = (0.75, 0.75)$		$(\tau_p, \tau_q) = (1.0, 1.0)$		$(\tau_p, \tau_q) = (1.5, 1.5)$	
	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD	Accept Len.	Avg. TVD
TV + K-SEQ	1.29 (±0.07)	0.2914	1.85 (±0.05)	0.4936	2.26 (±0.04)	0.6219	2.77 (±0.02)	0.7512
LV + K-SEQ	1.50 (±0.07)	0.2920	2.10 (±0.05)	0.4936	2.51 (±0.03)	0.6221	2.98 (±0.02)	0.7515
TV + RRS	1.26 (±0.07)	0.2918	1.79 (±0.05)	0.4939	2.18 (±0.04)	0.6221	2.68 (±0.02)	0.7514
LV + RRS	1.44 (±0.07)	0.2918	2.01 (±0.05)	0.4940	2.41 (±0.03)	0.6223	2.88 (±0.02)	0.7515
TV + GR	1.36 (±0.07)	0.2913	1.98 (±0.06)	0.4935	2.44 (±0.04)	0.6219	2.99 (±0.02)	0.7512
LV + GR	1.57 (±0.07)	0.2917	2.22 (±0.05)	0.4936	2.66 (±0.03)	0.6216	3.14 (±0.02)	0.7513
TRV	1.46 (±0.07)	0.2917	2.04 (±0.05)	0.4937	2.45 (±0.03)	0.6223	2.93 (±0.02)	0.7512
GBV	1.58 (±0.07)	0.2917	2.20 (±0.05)	0.4937	2.60 (±0.03)	0.6220	2.98 (±0.02)	0.7513
Baseline	–	0.2913	–	0.4933	–	0.6217	–	0.7511

H. Additional Details of LLM Decoding Experiments

Task. The Spec-Bench dataset (Xia et al., 2024) comprises 480 queries, randomly sampled from six different domains (80 queries each) of real-world natural language benchmarks:

1. Multi-Turn Conversation: MT-Bench (Zheng et al., 2023);
2. Translation: WMT14 DE-EN (Bojar et al., 2014);
3. Summarization: CNN/Daily Mail (Nallapati et al., 2016);
4. Question Answering: Natural Questions (Kwiatkowski et al., 2019);
5. Mathematical Reasoning: GSM8K (Cobbe et al., 2021);
6. Retrieval-Augmented Generation: Natural Questions (Kwiatkowski et al., 2019) + top-5 documents retrieved from DPR (Karpukhin et al., 2020).

In Tabs. 16, 17, 18 and 19, we display all experimental results summarized for each of six domains.

Draft and Target Models. We use pretrained weights of Llama3 series models (Grattafiori et al., 2024). In particular, following a similar experimental setting used by Weng et al. (2025), we use Llama3.1-8B-Instruct[5] as a target model and Llama3.2-1B-Instruct[6] as a draft model. We set the generation configurations for both the draft and target models to `top_p=0.9` and `temperature=0.6`, using the default parameters in `generation_config.json`.

Implementation Details. We mostly adopt the draft generation, verification, and metric computation pipeline from the PyTorch implementation of EAGLE (Li et al., 2024a), contained in the GitHub repository of Spec-Bench.[7] To reduce the total runtime, we use a small value of `max_new_tokens=128`: we stop decoding when the generated sequence exceeds 128 tokens. Since the Llama3-based draft model is not tailored for EAGLE-based draft tree generation, we slightly modified the draft construction code so that the draft trees are generated from the outputs of several parallel forward passes of the draft model, rather than exploiting hidden features as in EAGLE's pipeline.

For each tree configuration (double-chain tree or complete binary tree with depth 4 or 5), we run experiments for all eight verification methods in $(\{\text{TV}, \text{LV}\} \times \{\text{K-SEQ}, \text{RRS}, \text{GR}\}) \cup \{\text{default}, \text{TRV}\}$, for 4 different random seeds $(0, 1, 2, 3)$. The method called 'default' is equivalent to TV + RRS in terms of acceptance rules but keeps the original implementation of RRS which sequentially determines acceptance (or rejection) of next-token candidate tokens, one by one; on the other hand, TV + RRS is the TV-lifted algorithm by converting RRS into an equivalent flow network, as we do in LV + RRS. Motivated by the experimental settings of Sun et al. (2024) and Thomas & Pal (2026b), we limit the number of child nodes at each draft tree node and set the top-k sampling parameter of the draft model to be $k = 20$ in order to encourage a favorable computation of single-step verifiers' flow-network formulations.

All verification rules are efficiently implemented with a PyTorch C++/CUDA extension: C++ handles the verification logic and launches CUDA kernels for expensive per-node probability computations for TV and LV.

Hardware Details. We run all experiments on a server with four RTX A6000 GPUs (each with 48GB VRAM) and two AMD EPYC 7763 64-core CPUs. We build four disjoint Docker containers, each with exclusive access to a single GPU and 32 CPU cores; each experiment runs in a single container. For a fixed tree configuration, all experiments across 8 verification methods and 4 random seeds are run within 13.45 GPU hours.

[5] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Released with Llama3.1 Community License.
[6] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct. Released with Llama3.2 Community License.
[7] https://github.com/hemingkx/Spec-Bench. Released with Apache-2.0 license.

Table 16. Spec-Bench results on Multi-Chain (4,2).

Method	Multi-Turn Conversation				Translation				Summarization			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		35.85 (±0.01)	+0.0%	1.00 (±0.00)		36.25 (±0.02)	+0.0%	1.00 (±0.00)		32.96 (±0.02)	+0.0%
TV + K-SEQ	3.61 (±0.01)		44.04 (±0.11)	+22.8%	3.15 (±0.05)		37.13 (±0.64)	+2.4%	3.34 (±0.01)		38.47 (±0.12)	+16.7%
LV + K-SEQ	3.66 (±0.00)	+1.39%	44.29 (±0.15)	+23.5%	3.17 (±0.03)	+0.63%	37.21 (±0.35)	+2.6%	3.38 (±0.02)	+1.20%	38.76 (±0.06)	+17.6%
TV + RRS	3.62 (±0.00)		44.22 (±0.06)	+23.4%	3.16 (±0.04)		37.48 (±0.53)	+3.4%	3.36 (±0.02)		38.86 (±0.16)	+17.9%
LV + RRS	3.64 (±0.02)	+0.55%	44.57 (±0.23)	+24.3%	3.15 (±0.05)	-0.32%	37.17 (±0.57)	+2.5%	3.39 (±0.02)	+0.89%	39.15 (±0.26)	+18.8%
TV + GR	3.63 (±0.02)		43.75 (±0.09)	+22.0%	3.18 (±0.04)		37.03 (±0.44)	+2.2%	3.38 (±0.02)		38.56 (±0.20)	+17.0%
LV + GR	3.68 (±0.02)	+1.38%	44.32 (±0.25)	+23.6%	3.15 (±0.02)	-0.94%	36.78 (±0.33)	+1.5%	3.38 (±0.01)	+0.00%	38.47 (±0.05)	+16.7%
default	3.64 (±0.01)		44.07 (±0.15)	+22.9%	3.17 (±0.02)		37.06 (±0.25)	+2.3%	3.35 (±0.02)		38.41 (±0.14)	+16.5%
TRV	3.67 (±0.01)		44.14 (±0.08)	+23.1%	3.22 (±0.04)		37.37 (±0.46)	+3.1%	3.36 (±0.01)		38.19 (±0.09)	+15.9%

Method	Question Answering				Mathematical Reasoning				Retrieval-Augmented Generation			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		36.46 (±0.01)	+0.0%	1.00 (±0.00)		36.27 (±0.01)	+0.0%	1.00 (±0.00)		31.65 (±0.07)	+0.0%
TV + K-SEQ	3.37 (±0.02)		41.17 (±0.24)	+12.9%	4.09 (±0.01)		50.27 (±0.25)	+38.6%	3.53 (±0.01)		37.35 (±0.19)	+18.0%
LV + K-SEQ	3.44 (±0.01)	+2.08%	41.82 (±0.16)	+14.7%	4.11 (±0.02)	+0.49%	50.24 (±0.23)	+38.5%	3.56 (±0.02)	+0.85%	37.43 (±0.17)	+18.3%
TV + RRS	3.39 (±0.03)		41.61 (±0.28)	+14.1%	4.09 (±0.01)		50.37 (±0.14)	+38.9%	3.53 (±0.03)		37.45 (±0.37)	+18.3%
LV + RRS	3.43 (±0.03)	+1.18%	42.08 (±0.33)	+15.4%	4.11 (±0.01)	+0.49%	50.63 (±0.23)	+39.6%	3.53 (±0.00)	+0.00%	37.51 (±0.13)	+18.5%
TV + GR	3.39 (±0.02)		41.08 (±0.19)	+12.7%	4.10 (±0.01)		49.87 (±0.15)	+37.5%	3.53 (±0.04)		37.10 (±0.32)	+17.2%
LV + GR	3.44 (±0.02)	+1.47%	41.42 (±0.12)	+13.6%	4.15 (±0.01)	+1.22%	50.28 (±0.18)	+38.6%	3.56 (±0.03)	+0.85%	37.14 (±0.31)	+17.3%
default	3.39 (±0.01)		41.23 (±0.17)	+13.1%	4.10 (±0.01)		49.95 (±0.24)	+37.7%	3.52 (±0.01)		36.86 (±0.15)	+16.5%
TRV	3.41 (±0.03)		41.02 (±0.33)	+12.5%	4.13 (±0.02)		49.97 (±0.21)	+37.8%	3.54 (±0.01)		36.92 (±0.13)	+16.6%

Table 17. Spec-Bench results on Complete-Tree (4,2).

Method	Multi-Turn Conversation				Translation				Summarization			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		35.85 (±0.01)	+0.0%	1.00 (±0.00)		36.25 (±0.02)	+0.0%	1.00 (±0.00)		32.96 (±0.02)	+0.0%
TV + K-SEQ	3.75 (±0.01)		45.10 (±0.20)	+25.8%	3.31 (±0.02)		38.14 (±0.22)	+5.2%	3.50 (±0.01)		39.07 (±0.17)	+18.5%
LV + K-SEQ	3.81 (±0.01)	+1.60%	45.56 (±0.26)	+27.1%	3.28 (±0.04)	-0.91%	37.80 (±0.50)	+4.3%	3.51 (±0.01)	+0.29%	39.07 (±0.17)	+18.5%
TV + RRS	3.77 (±0.01)		45.75 (±0.12)	+27.6%	3.33 (±0.03)		38.75 (±0.47)	+6.9%	3.49 (±0.02)		39.38 (±0.26)	+19.5%
LV + RRS	3.78 (±0.02)	+0.27%	46.02 (±0.27)	+28.4%	3.30 (±0.04)	-0.90%	38.56 (±0.40)	+6.4%	3.53 (±0.00)	+1.15%	39.83 (±0.15)	+20.9%
TV + GR	3.84 (±0.00)		46.38 (±0.18)	+29.4%	3.30 (±0.01)		38.25 (±0.24)	+5.5%	3.55 (±0.02)		39.84 (±0.19)	+20.9%
LV + GR	3.84 (±0.01)	+0.00%	46.37 (±0.31)	+29.4%	3.34 (±0.03)	+1.21%	38.63 (±0.40)	+6.6%	3.58 (±0.00)	+0.85%	40.01 (±0.06)	+21.4%
default	3.78 (±0.01)		45.47 (±0.14)	+26.8%	3.28 (±0.02)		37.91 (±0.32)	+4.6%	3.52 (±0.02)		39.31 (±0.20)	+19.3%
TRV	3.80 (±0.00)		44.75 (±0.14)	+24.8%	3.30 (±0.03)		37.18 (±0.46)	+2.6%	3.54 (±0.02)		38.71 (±0.15)	+17.5%

Method	Question Answering				Mathematical Reasoning				Retrieval-Augmented Generation			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		36.46 (±0.01)	+0.0%	1.00 (±0.00)		36.27 (±0.01)	+0.0%	1.00 (±0.00)		31.65 (±0.07)	+0.0%
TV + K-SEQ	3.50 (±0.01)		41.88 (±0.10)	+14.9%	4.21 (±0.02)		50.96 (±0.35)	+40.5%	3.62 (±0.03)		37.41 (±0.22)	+18.2%
LV + K-SEQ	3.55 (±0.02)	+1.43%	42.30 (±0.29)	+16.0%	4.27 (±0.02)	+1.43%	51.37 (±0.22)	+41.6%	3.65 (±0.03)	+0.83%	37.64 (±0.36)	+18.9%
TV + RRS	3.51 (±0.03)		42.56 (±0.39)	+16.7%	4.24 (±0.02)		51.78 (±0.25)	+42.8%	3.65 (±0.02)		38.00 (±0.08)	+20.1%
LV + RRS	3.56 (±0.00)	+1.42%	43.16 (±0.21)	+18.4%	4.26 (±0.02)	+0.47%	52.01 (±0.16)	+43.4%	3.70 (±0.01)	+1.37%	38.47 (±0.15)	+21.6%
TV + GR	3.61 (±0.01)		43.46 (±0.21)	+19.2%	4.29 (±0.02)		51.99 (±0.32)	+43.3%	3.70 (±0.01)		38.52 (±0.13)	+21.7%
LV + GR	3.64 (±0.01)	+0.83%	43.65 (±0.17)	+19.7%	4.35 (±0.01)	+1.40%	52.70 (±0.10)	+45.3%	3.71 (±0.03)	+0.27%	38.30 (±0.23)	+21.0%
default	3.52 (±0.02)		42.10 (±0.15)	+15.5%	4.21 (±0.01)		50.84 (±0.16)	+40.2%	3.64 (±0.04)		37.37 (±0.34)	+18.1%
TRV	3.52 (±0.01)		41.26 (±0.14)	+13.2%	4.25 (±0.01)		50.62 (±0.09)	+39.6%	3.69 (±0.01)		37.53 (±0.12)	+18.6%

Table 18. Spec-Bench results on Multi-Chain (5,2).

Method	Multi-Turn Conversation				Translation				Summarization			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		35.85 (±0.01)	+0.0%	1.00 (±0.00)		36.25 (±0.02)	+0.0%	1.00 (±0.00)		32.96 (±0.02)	+0.0%
TV + K-SEQ	3.97 (±0.02)		42.21 (±0.30)	+17.7%	3.35 (±0.04)		34.33 (±0.29)	-5.3%	3.63 (±0.01)		36.52 (±0.19)	+10.8%
LV + K-SEQ	4.05 (±0.01)	+2.02%	42.43 (±0.11)	+18.3%	3.47 (±0.04)	+3.58%	34.94 (±0.26)	-3.6%	3.71 (±0.02)	+2.20%	37.08 (±0.11)	+12.5%
TV + RRS	3.99 (±0.02)		42.32 (±0.37)	+18.0%	3.41 (±0.04)		34.75 (±0.46)	-4.1%	3.65 (±0.01)		36.80 (±0.22)	+11.6%
LV + RRS	4.05 (±0.02)	+1.50%	42.95 (±0.18)	+19.8%	3.45 (±0.06)	+1.17%	35.26 (±0.65)	-2.7%	3.70 (±0.03)	+1.37%	37.15 (±0.23)	+12.7%
TV + GR	4.04 (±0.02)		42.44 (±0.17)	+18.4%	3.38 (±0.05)		34.38 (±0.63)	-5.2%	3.67 (±0.02)		36.73 (±0.22)	+11.4%
LV + GR	4.09 (±0.03)	+1.24%	42.62 (±0.19)	+18.9%	3.45 (±0.05)	+2.07%	34.61 (±0.41)	-4.5%	3.71 (±0.01)	+1.09%	36.75 (±0.11)	+11.5%
default	4.03 (±0.03)		42.09 (±0.28)	+17.4%	3.44 (±0.02)		34.58 (±0.20)	-4.6%	3.67 (±0.02)		36.50 (±0.29)	+10.8%
TRV	4.08 (±0.01)		42.22 (±0.17)	+17.8%	3.48 (±0.05)		34.74 (±0.40)	-4.2%	3.69 (±0.03)		36.40 (±0.25)	+10.5%

Method	Question Answering				Mathematical Reasoning				Retrieval-Augmented Generation			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		36.46 (±0.01)	+0.0%	1.00 (±0.00)		36.27 (±0.01)	+0.0%	1.00 (±0.00)		31.65 (±0.07)	+0.0%
TV + K-SEQ	3.71 (±0.03)		39.32 (±0.15)	+7.8%	4.66 (±0.01)		49.55 (±0.27)	+36.6%	3.88 (±0.03)		35.69 (±0.35)	+12.8%
LV + K-SEQ	3.73 (±0.02)	+0.54%	39.24 (±0.11)	+7.6%	4.71 (±0.02)	+1.07%	49.87 (±0.13)	+37.5%	3.93 (±0.04)	+1.29%	36.03 (±0.27)	+13.8%
TV + RRS	3.68 (±0.01)		39.06 (±0.19)	+7.1%	4.66 (±0.02)		49.63 (±0.37)	+36.8%	3.91 (±0.04)		35.91 (±0.25)	+13.5%
LV + RRS	3.71 (±0.02)	+0.82%	39.26 (±0.18)	+7.7%	4.70 (±0.02)	+0.86%	49.98 (±0.18)	+37.8%	3.92 (±0.04)	+0.26%	36.07 (±0.24)	+14.0%
TV + GR	3.70 (±0.01)		38.99 (±0.24)	+6.9%	4.69 (±0.02)		49.45 (±0.29)	+36.3%	3.96 (±0.03)		36.13 (±0.31)	+14.2%
LV + GR	3.81 (±0.03)	+2.97%	39.67 (±0.27)	+8.8%	4.76 (±0.01)	+1.49%	49.78 (±0.08)	+37.3%	4.01 (±0.05)	+1.26%	36.20 (±0.32)	+14.4%
default	3.68 (±0.04)		38.46 (±0.29)	+5.5%	4.65 (±0.02)		48.74 (±0.24)	+34.4%	3.90 (±0.05)		35.56 (±0.49)	+12.4%
TRV	3.75 (±0.03)		39.10 (±0.29)	+7.2%	4.66 (±0.02)		48.82 (±0.26)	+34.6%	3.94 (±0.03)		35.79 (±0.24)	+13.1%

Table 19. Spec-Bench results on Complete-Tree (5,2).

Method	Multi-Turn Conversation				Translation				Summarization			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		35.85 (±0.01)	+0.0%	1.00 (±0.00)		36.25 (±0.02)	+0.0%	1.00 (±0.00)		32.96 (±0.02)	+0.0%
TV + K-SEQ	4.20 (±0.02)		42.24 (±0.20)	+17.8%	3.56 (±0.02)		34.56 (±0.17)	-4.7%	3.87 (±0.02)		34.33 (±0.13)	+4.2%
LV + K-SEQ	4.27 (±0.02)	+1.67%	42.40 (±0.17)	+18.3%	3.61 (±0.02)	+1.40%	34.54 (±0.22)	-4.7%	3.90 (±0.02)	+0.78%	34.16 (±0.13)	+3.6%
TV + RRS	4.22 (±0.02)		42.92 (±0.30)	+19.7%	3.59 (±0.02)		35.13 (±0.40)	-3.1%	3.89 (±0.02)		34.77 (±0.21)	+5.5%
LV + RRS	4.28 (±0.02)	+1.42%	43.41 (±0.22)	+21.1%	3.59 (±0.03)	+0.00%	35.20 (±0.31)	-2.9%	3.90 (±0.01)	+0.26%	34.81 (±0.06)	+5.6%
TV + GR	4.31 (±0.01)		43.19 (±0.21)	+20.5%	3.60 (±0.02)		34.73 (±0.09)	-4.2%	3.96 (±0.01)		35.03 (±0.16)	+6.3%
LV + GR	4.34 (±0.02)	+0.70%	43.22 (±0.30)	+20.6%	3.70 (±0.03)	+2.78%	35.35 (±0.25)	-2.5%	4.00 (±0.01)	+1.01%	35.09 (±0.15)	+6.5%
default	4.22 (±0.02)		42.22 (±0.18)	+17.8%	3.61 (±0.05)		34.66 (±0.50)	-4.4%	3.87 (±0.01)		34.37 (±0.10)	+4.3%
TRV	4.27 (±0.01)		41.39 (±0.10)	+15.5%	3.61 (±0.07)		33.36 (±0.64)	-8.0%	3.90 (±0.02)		33.40 (±0.16)	+1.4%

Method	Question Answering				Mathematical Reasoning				Retrieval-Augmented Generation			
	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup	Block Eff.	$\Delta_{\text{BE}}^{(\text{LV, TV})}$	Token/s	Speedup
Baseline	1.00 (±0.00)		36.46 (±0.01)	+0.0%	1.00 (±0.00)		36.27 (±0.01)	+0.0%	1.00 (±0.00)		31.65 (±0.07)	+0.0%
TV + K-SEQ	3.89 (±0.03)		39.19 (±0.29)	+7.5%	4.87 (±0.04)		49.27 (±0.26)	+35.9%	4.03 (±0.02)		33.44 (±0.24)	+5.6%
LV + K-SEQ	3.95 (±0.02)	+1.54%	39.19 (±0.19)	+7.5%	4.92 (±0.02)	+1.03%	48.98 (±0.28)	+35.0%	4.11 (±0.05)	+1.99%	33.53 (±0.22)	+5.9%
TV + RRS	3.86 (±0.02)		39.28 (±0.27)	+7.7%	4.86 (±0.04)		49.60 (±0.52)	+36.8%	4.07 (±0.02)		33.96 (±0.16)	+7.3%
LV + RRS	3.95 (±0.03)	+2.33%	40.19 (±0.31)	+10.2%	4.92 (±0.01)	+1.23%	50.23 (±0.33)	+38.5%	4.13 (±0.05)	+1.47%	34.29 (±0.29)	+8.3%
TV + GR	3.98 (±0.03)		39.91 (±0.26)	+9.5%	4.99 (±0.04)		50.31 (±0.38)	+38.7%	4.08 (±0.04)		33.82 (±0.17)	+6.9%
LV + GR	4.06 (±0.03)	+2.01%	40.34 (±0.16)	+10.6%	5.04 (±0.01)	+1.00%	50.62 (±0.22)	+39.6%	4.15 (±0.03)	+1.72%	34.08 (±0.20)	+7.7%
default	3.94 (±0.06)		39.31 (±0.50)	+7.8%	4.90 (±0.01)		49.27 (±0.04)	+35.8%	4.11 (±0.07)		34.08 (±0.37)	+7.7%
TRV	3.96 (±0.05)		38.19 (±0.46)	+4.8%	4.93 (±0.01)		48.54 (±0.20)	+33.8%	4.07 (±0.03)		32.76 (±0.10)	+3.5%