Merged-Tree Dual Drafting: Verify the Union Once

February 25, 2026

One sentence. Instead of choosing one draft head per step, we take both heads' speculative trees, splice them into one combined tree with a shared root, run the verifier once on that union of candidates, and accept the best verifier-consistent prefix exactly as in single-head speculative decoding.

1 Objects and Notation

We generate tokens for prompt x. Let $y_{1:t}$ be the currently accepted continuation.

- **Verifier.** Distribution $p_V(\cdot \mid x, y_{1:t})$.
- **Two draft heads.** $p_{D_1}(\cdot \mid x, y_{1:t})$ and $p_{D_2}(\cdot \mid x, y_{1:t})$.
- **Root token.** A token s_{t+1} drawn from the verifier under the chosen decoding policy (greedy or sampling).

At each speculative step, both draft heads produce a rooted tree of candidate continuations after the same root token s_{t+1}.

2 Two Draft Trees

Head $h \in \{1, 2\}$ proposes a rooted tree $\mathcal{T}_h$ with:

- a shared root token s_{t+1},
- a finite retained node set S_h of draft tokens (a budgeted subset),
- a set of root-to-leaf paths $\mathcal{P}_h$ extracted from the retained structure.

Each path $p \in \mathcal{P}_h$ corresponds to a token sequence

$$u^{(h,p)}_{1:\ell(p)} = \left(u^{(h,p)}_1, \ldots, u^{(h,p)}_{\ell(p)}\right).$$

The full candidate continuation represented by that path is

$$c^{(h,p)} = (s_{t+1}, u^{(h,p)}_{1:\ell(p)}).$$

3 The Merge: Build a Union Candidate Set

Merged-tree dual drafting does not choose between heads. It forms a combined structure $\mathcal{T}_\cup$ whose candidate paths are simply the union:

$$\mathcal{C}_\cup = \{c^{(1,p)} : p \in \mathcal{P}_1\} \cup \{c^{(2,p)} : p \in \mathcal{P}_2\}.$$

Key idea. The verifier is run once on $\mathcal{T}_\cup$, so the verification cost per step is comparable to verifying a single tree with the same total retained-token budget, while the candidate set covers both draft heads.

3.1 Structural constraints for a valid merged tree

To verify all candidates in one forward pass, the combined object must be representable as a single tree-shaped batch with a consistent attention mask and position indices. The merge uses three invariants:

- **Shared root.** Both trees begin with the same s_{t+1}, so the merged tree has one root.
- **Disjoint subtrees under the root.** Nodes originating from head 1 and head 2 do not point to each other; they only share the root.
- **Index remapping.** When concatenating node lists, head-2 node indices are offset so that all parent pointers and path indices remain correct in the merged coordinate system.

4 Merging With Attention and Positions

The point of merging is not just set union; it is the ability to evaluate *all* candidates in one verifier forward pass. This requires three bookkeeping objects for the verifier:

- a merged list of candidate tokens (the nodes),
- a merged tree attention mask,
- merged position identifiers.

4.1 Indexing and Remapping

Assume each head produces an ordered list of nodes including the shared root:

$$\mathcal{N}_h = (0, 1, 2, \ldots, n_h),$$

where node 0 denotes the root token s_{t+1} and nodes $1..n_h$ are the retained draft tokens from head h. Each head also defines a parent pointer $\pi_h(i)$ for nodes $i \in \{1, \ldots, n_h\}$ with $\pi_h(i) \in \{0, \ldots, n_h\}$.

The merged node set is indexed as

$$\mathcal{N}_\cup = (0, 1, \ldots, n_1, n_1 + 1, \ldots, n_1 + n_2),$$

constructed by concatenating head 1's non-root nodes with head 2's non-root nodes. Define the embedding map from head-2 indices to merged indices:

$$\varphi(0) = 0, \qquad \varphi(i) = n_1 + i \text{ for } i \in \{1, \ldots, n_2\}.$$

Head-1 nodes keep their indices unchanged, while every head-2 node $i > 0$ is shifted by n_1. Parent pointers are remapped accordingly:

$$\pi_\cup(j) = \begin{cases} \pi_1(j), & j \in \{1, \ldots, n_1\}, \\ \varphi(\pi_2(j - n_1)), & j \in \{n_1 + 1, \ldots, n_1 + n_2\}. \end{cases}$$

This preserves tree ancestry while preventing any cross-links between the two subtrees.

4.2 Tree Attention Mask

Verifier evaluation treats the merged nodes as one packed sequence of length $N = 1 + n_1 + n_2$ (including the root). We define a binary mask $M_\cup \in \{0, 1\}^{N \times N}$ so that each node attends only to the root and its own ancestors.

First define the ancestor set in the merged tree:

$$\mathrm{Anc}_\cup(i) = \{i, \pi_\cup(i), \pi_\cup(\pi_\cup(i)), \ldots, 0\}.$$

Then the tree attention mask is

$$M_\cup[i, j] = \mathbf{1}[j \in \mathrm{Anc}_\cup(i)].$$

Equivalently, $M_\cup$ is block-structured:

$$M_\cup = \begin{pmatrix} 1 & 0 & 0 \\ \mathbf{1} & M_1 & 0 \\ \mathbf{1} & 0 & M_2 \end{pmatrix},$$

where M_1 is head 1's within-tree ancestor mask on nodes $1..n_1$, M_2 is head 2's within-tree ancestor mask on nodes $1..n_2$ (after reindexing), and $\mathbf{1}$ indicates that every non-root node attends to the shared root. The off-diagonal blocks are zero, encoding the disjointness of the two subtrees under the root.

In a transformer implementation, this tree mask is applied as a causal-style restriction inside self-attention for the verifier pass over the packed nodes, so that computations factor along the tree rather than mixing tokens from unrelated branches.

4.3 Position Identifiers

All nodes in the merged tree must be assigned position identifiers that are consistent with the underlying prefix length t. A simple scheme is *depth positions*:

$$\mathrm{pos}(0) = 0, \qquad \mathrm{pos}(i) = \mathrm{pos}(\pi_\cup(i)) + 1.$$

The verifier then uses absolute positions

$$\mathrm{abspos}(i) = t + \mathrm{pos}(i),$$

so the root token sits at $t + 0$ relative to the prefix boundary and every child advances by one position along its path. Because both draft trees share the root and use the same depth semantics, depth-based positions align naturally across the two subtrees.

4.4 Path Retrieval as a Matrix

To evaluate acceptance, it is useful to represent the candidate set as a matrix of root-to-leaf paths with padding. Let $R_\cup \in \{-1, 0, 1, \ldots, N-1\}^{L \times D}$ list L candidate paths (rows), each of maximum depth D (columns), with -1 padding for missing entries. Each row begins with the root index 0 and then follows parent pointers down to a leaf in one subtree. The remapping φ guarantees that head-1 and head-2 paths coexist without index collisions.

5 Verifier Scoring on the Union

Let the verifier assign logits to each node position along each candidate path in $\mathcal{C}_\cup$. This is a batched computation: the verifier sees the prefix $y_{1:t}$ plus a tree of candidate tokens. For each candidate $c \in \mathcal{C}_\cup$ and each depth j, the verifier defines a conditional probability:

$$p_V\big(c_j \mid x, y_{1:t}, c_{<j}\big).$$

6 Acceptance: Same Rule as Standard Speculative Decoding

From the merged verifier logits, the algorithm selects:

- a best candidate $c^\star \in \mathcal{C}_\cup$,
- an accepted prefix length $a \geq 0$ along $c^\star$ beyond the root token (so accepted tokens include $(s_{t+1}, c^\star_{2:a+1})$).

6.1 Greedy acceptance (temperature $= 0$)

Let $\hat{z}_{t+j}$ be the verifier argmax token at position $t + j$. For a candidate c define

$$m_j^{(c)} = \mathbf{1}[c_j = \hat{z}_{t+j-1}], \quad j \geq 2,$$

where $c_1 = s_{t+1}$ is the root token and indexing is aligned so c_2 corresponds to the first draft token after the root. The accepted length is the longest matching prefix:

$$a^{(c)} = \max \left\{ r : \prod_{j=2}^{r+1} m_j^{(c)} = 1 \right\}.$$

Choose $c^\star$ that maximizes $a^{(c)}$ and set $a = a^{(c^\star)}$.

6.2 Sampling acceptance (temperature > 0)

Under sampling, the verifier distribution is transformed by a decoding operator $\mathcal{G}$ (temperature, top-p, top-k). Along a candidate, acceptance proceeds sequentially with stochastic accept/reject against the transformed verifier probabilities, renormalizing upon rejections and sampling the next appended token from the updated distribution. The output is a best candidate $c^\star$ and an accepted length a.

7 Commit and State Flow

Once $(c^\star, a)$ are chosen, we commit:

$$y_{1:t} \leftarrow \big(y_{1:t},\, c^\star_{1:a+1}\big).$$

The verifier's KV cache is updated so that future steps reuse computation for the newly extended prefix.

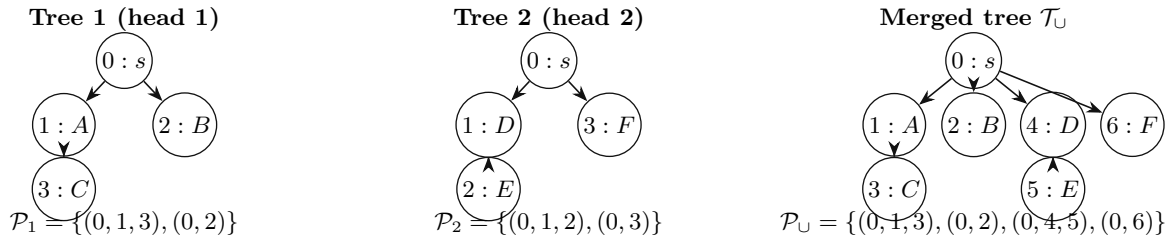
Both draft heads then advance using the accepted hidden states produced by the verifier for the committed tokens. So at the next step, both heads are conditioned on the same $y_{1:t}$ again.

8 Merged Tree vs. Step-Level Routing

Step-level routing chooses one head and verifies one tree. Merged-tree dual drafting verifies a union tree.

- **Coverage.** Merged-tree includes candidates from both heads at the same step.
- **Draft overhead.** Both methods require generating both draft trees; routing also computes a head-selection score, while merging builds a combined index/mask structure.
- **Verifier work.** Routing verifies one head's candidates; merging verifies the union in a single pass at a larger candidate width.

9 One Diagram



Index remapping used by the merge. Let n_1 be the number of non-root nodes in Tree 1. Keep Tree 1 indices unchanged. Map Tree 2 indices by $\varphi(0) = 0$ and $\varphi(i) = n_1 + i$ for $i > 0$. In this example $n_1 = 3$, so $1 \mapsto 4$, $2 \mapsto 5$, $3 \mapsto 6$. The merged attention mask allows each node to attend only to itself and its ancestors, so the two subtrees remain disjoint except for the shared root.

Figure 1: A worked example of merged-tree dual drafting with shared root token and index remapping for the second head.