APPENDIX

A EXPERIMENTAL RESULTS

A.1 MARKOV PROBES

We design a toy Markov transition task to test whether attention can recover structured dynamics. A sparse block-graph is converted into a row-stochastic matrix $P^\star$. Each training sample is a sequence `[0..V-1, i]` where the last token i is the query, and the target distribution is $P^\star_{i,:}$. We compare a parameter-matched vanilla Transformer and an S-Former layer, training with KL divergence between the query's attention row and $P^\star_{i,:}$. Evaluation reports the mean ℓ_1 distance between learned $\hat{P}$ and $P^\star$.

Algorithm 1 Markov Attention Matching

```
Generate sparse graph → P*
for epoch do
    for query i do
        Input [0..V-1, i] → model
        Extract attention row at last position
        Minimize KL(P*_{i,:} ‖ p̂_{i,:})
Eval: average rows → P̂, compute L1(P̂, P*)
```

A.2 DYCK PROBES

We evaluate S-Former on Dyck language completion tasks as a canonical probe for hierarchical generalization. **Data generation.** Sequences are sampled with three bracket types `()[]{}` using a stack-based generator. Each completion sample is split into *(input, target)*, where the model sees <sos>+input and must generate the target. Depth-controlled sets (d = 3–8) are also constructed. Invalid samples are produced by deleting, replacing, or inserting random brackets.

Algorithm 2 Dyck Completion Training & Evaluation

```
Input: Generator G, model M, config C
Generate train/val/test via G
for epoch = 1..50 do
    for batch in train do
        X ← sequence[:-1], Y ← sequence[1:]
        mask ← 1 on target positions
        logits ← M(X)
        L ← CE(logits, Y, ignore=<pad>)
        loss ← Σ(L · mask) / Σ(mask)
        Update M with AdamW, gradient clip, scheduler
Evaluation (greedy): given <sos>+input, generate until <eos> or max length; filter to ().
Compute Exact-Match, Structural Acc, Valid-but-not-Exact, Syntax Error.
Extrapolation: repeat for lengths {64, 80, 100} and depths {3, ..., 8}.
```

Training. Models are trained with AdamW (lr $= 1e{-}4$), batch size 32, 50 epochs, and 3 warmup epochs followed by cosine decay. The loss is masked to only target tokens.

Evaluation metrics.

- Exact-Match Accuracy
- Structural Accuracy (valid Dyck sequence check)
- Valid-but-not-Exact rate
- Syntax Error rate = $1-$ Structural Accuracy

Extrapolation. We evaluate on extended lengths ($L \in \{64, 80, 100\}$) and depths (d = 3–8). For S-Former, we additionally test sensitivity by corrupting the structural stream (permutation or noise).

A.3 JSON PROBES

We evaluate S-Former on JSON serialization and completion to test hierarchical and schema-aware generalization.

Data generation. We synthesize JSON objects/arrays with controlled nesting depth and width. Keys are sampled from a fixed vocabulary (e.g., {"name","id","value","items","meta","ts"}), values are strings, integers, booleans, nulls, or recursively nested structures. Each example is converted to a compact, whitespace-free string; we then split it into *(input, target)* such that the model sees <sos>+input and must generate the remaining *target*. Negative (invalid) samples are created by bracket/quote corruption, missing commas/colons, or reordering that breaks JSON syntax. A validator based on `json.loads` defines structural validity.

Controls. We construct depth-controlled sets ($d = 2$–6), width-controlled sets (avg keys per object $w = 2$–8), and length buckets (token length $L \in \{128, 256, 512\}$). Key-set splits ensure that some keys only appear at test time to probe schema extrapolation.

Training. AdamW (lr $= 1e{-}4$), batch size 32, 50 epochs, 3 warmup epochs then cosine decay. Loss is masked to target tokens only.

Evaluation metrics.

- **Structural Validity**: passes `json.loads` (*valid JSON*).
- **Field F1**: compare parsed objects on a normalized key set (micro-F1 over present/absent key paths).
- **Syntax Error rate** $= 1-$ Structural Validity.

Model	Struct (loose)	Depth (loose)
Standard Transformer	0.845	0.095
RoPE Transformer	0.857	0.125
S-Former (`pure`)	0.903	0.035
S-Former (`fused`)	0.806	0.100
S-Former (`dynamic`)	**0.953**	**0.055**

Algorithm 3 JSON Completion Training & Evaluation

Input: JSON generator G, model M, config C
Sample JSON trees with depth/width controls; stringify without spaces
Split each string into *(input, target)*; build sequences with <sos>
for epoch = 1..50 **do**
  for batch in train **do**
    $X \leftarrow$ sequence[:-1], $Y \leftarrow$ sequence[1:]
    $mask \leftarrow 1$ on target positions; 0 elsewhere
    $logits \leftarrow M(X)$
    $L \leftarrow$ CE($logits, Y$, ignore=<pad>)
    $loss \leftarrow \sum(L \cdot mask) / \sum(mask)$
    Update M (AdamW, grad clip, scheduler)
Evaluation (greedy): given <sos>+input, generate until <eos> or max length
Structural Validity: `try: json.loads(generated)` $\Rightarrow$ valid/invalid
Field F1: parse gold/pred JSON; flatten to key-path sets; compute micro-F1
Report Exact-Match, Structural Validity, Field F1, Syntax Error
Extrapolation: repeat across depths $\{2..6\}$, lengths $\{128, 256, 512\}$, and unseen key-sets
(S-Former only) optionally corrupt structural stream (permute/noise) at eval to measure sensitivity

A.4 WIKITEXT-103 SETUP (ATTENTION FUSION)

Architecture. Baseline: a Pre-LN Transformer with RoPE positional encoding (base = 50k) applied to Q, K; $d_{\text{model}} = 256$, $n_{\text{layers}} = 4$, $n_{\text{heads}} = 8$, $d_{\text{ff}} = 1024$, dropout=0.1. S-Former (*attention*

fusion): replace selected layers (default: all four) with structural-fusion blocks; the fused representation feeds Q, K while V is taken from the content path. All other components remain identical.

Training. Models are trained only on 256-token windows with a sliding window (stride 64; 75% overlap). We use AdamW ($1e{-4}$), 2k warmup followed by cosine decay, batch size 64, label smoothing 0.03, and gradient clipping at 1.0.

Evaluation. We evaluate on the standard WikiText-103 test split and extrapolate to long contexts up to $L = 40{,}960$, reporting PPL/BPB and degradation relative to $L = 256$.

Note. This subsection reports the *attention fusion* integration. The *bias injection* variant uses the same data, model size, and schedule; only the integration mechanism differs (see the next subsection).

A.5 WikiText-103 Setup (Bias Injection)

Architecture. The baseline is a 4-layer Pre-LN Transformer (256 hidden size, 8 heads, 1024 FFN, dropout 0.1) with RoPE on queries and keys. Bias-injection S-Former augments each layer with a lightweight structural stream implemented as a GRU pathway. The structural state is added as a gated bias into hidden activations before both attention and feedforward blocks. Gating values are regularized to remain within a stable range, with a simple warm-up schedule applied during training.

Training. Both baseline and S-Former are trained on 256-token windows with stride 64, using AdamW (lr 1×10^{-4}, 2k warmup, cosine decay), batch size 64, label smoothing 0.03, and gradient clipping at 1.0.

Evaluation. PPL/BPB are reported on the WikiText-103 test split for $L = 256$ up to 40k tokens, with smaller batch sizes at long contexts. We also report gate statistics (mean α and near-saturation ratios).

Contrast. Unlike *attention fusion*, which mixes content and structure into Q, K, bias injection preserves the content stream and injects g_t as an additive memory bias at attention and FFN inputs.

B Implementation Details for LAMBADA and PG-19

Common Training Framework. Both LAMBADA and PG-19 experiments share the same implementation backbone. We use a 4-layer Pre-LN Transformer baseline ($d_{\text{model}} = 256$, $n_{\text{heads}} = 8$, $d_{\text{ff}} = 1024$, dropout $= 0.1$) with RoPE (base $= 10^5$) applied to queries and keys. For S-Former, we replace designated layers with structural blocks (GRU-based structural stream, bias injection, and α-gating with warmup scheduling). Training uses AdamW with cosine decay and 2k warmup steps, batch size 16–32, stride $\approx L/4$, label smoothing 0.03–0.05, gradient clipping 1.0, and gate warmup 4k steps with τ annealed from $4 \to 2$. All runs are trained for 5–10 epochs on consumer GPUs (A100), and we report mean values across multiple seeds when available.

Evaluation Metrics. Perplexity (PPL) is computed from negative log-likelihood; Bits-per-byte (BPB) is also reported for extrapolation experiments. Accuracy (ACC) differs slightly:

- **LAMBADA**: last-token accuracy, i.e., whether the model predicts the final word correctly.
- **PG-19**: token-level accuracy, i.e., the fraction of all next-token predictions that are correct.

Dataset Handling. Both datasets are loaded locally when possible, falling back to HuggingFace repositories. Sliding-window segmentation is applied during training. For evaluation:

- **LAMBADA**: Only samples whose target is a single token are retained (standard preprocessing). Evaluation is performed at sequence lengths 512–4096.
- **PG-19**: Trained and evaluated at length 4096 tokens, without overlap (fixed blocks). This reflects the book-level nature of PG-19.

Reproducibility. We release both scripts as `lambada_bias.py` and `pg19_bias.py`. The codebases are identical up to ∼95%, differing only in dataset loader and accuracy metric. For transparency, we provide both scripts in the repository, but summarize them here in unified form.

B.1 Supplementary Probe Analyses (Bias Injection)

We provide definitions of the probes used to analyze the bias stream $m_t = \alpha_t g_t$ in the bias injection variant.

- **Relative strength.** Ratio of bias to content norm:
$$r_t = \frac{\|m_t\|}{\|x_t\| + \varepsilon}.$$
- **Orthogonality.** Cosine-based independence measure:
$$\mathrm{Orth}(t) = 1 - \left|\langle \hat{m}_t, \hat{x}_t \rangle\right|.$$
- **Effective rank.** From SVD of bias vectors across tokens, the smallest k explaining 95% variance.
- **Temporal consistency.** Cosine similarity between consecutive bias vectors:
$$\mathrm{TempCons} = \frac{1}{T-1} \sum_{t=1}^{T-1} \langle \hat{m}_t, \hat{m}_{t+1} \rangle.$$
- **Representation shift.** Change of hidden states with and without bias:
$$\Delta h_t = \frac{\|h_t^{\text{biased}} - h_t^{\text{orig}}\|}{\|h_t^{\text{orig}}\| + \varepsilon}.$$
- **Spectral analysis.** Given eigenvalues $\{\lambda_i\}$ of the attention matrix:
$$\mathrm{EffDim} = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2}, \quad H = -\sum_i p_i \log(p_i + \varepsilon), \quad p_i = \frac{\lambda_i}{\sum_j \lambda_j}.$$
Here EffDim is effective dimensionality, and H is spectral entropy.

Across layers, these probes show that the bias stream remains nearly orthogonal (0.92–0.95), high-rank ($\sim$200/256), and temporally smooth (0.34$\rightarrow$0.60), supporting its interpretation as a structural anchor rather than a compressed memory cache.