STATIC-SOURCE RECURRENT ATTENTION: TIME, MEMORY, AND PRECISION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study recurrent attention with immutable, position-locally encoded inputs and finite-precision queries. The model captures a class of iterative latent readers, including local-input configurations of Perceiver with a fixed source-kernel library. We prove two communication bounds: a cumulative query bound and a finite response-table bound whose size is independent of recurrent time and mutable memory. For set disjointness on two N-bit blocks, these imply $TM = \Omega_\varepsilon(N)$ and $P = \Omega_\varepsilon(\log N)$ for any fixed worst-case error $\varepsilon < 1/2$. Thus additional recurrent computation cannot compensate for sublogarithmic source precision. A matching construction achieves $P = \Theta(\log N)$ and $TM = O(N \log N)$. Complementing the lower bounds, we construct a fixed rational-weight recurrent-attention program that simulates a τ-step, S-space Turing machine in $O(\tau + 1)$ rounds and $O(S + p)$ mutable bits using $p = \Theta(\log(n + S + 2))$ precision. A floating-point repair operation restores approximate softmax reads exactly before each state update, preventing error accumulation over time. An end-to-end finite-precision implementation matches independent Turing-machine traces for up to 10,010 transitions.

1 INTRODUCTION

Transformer computation is constrained by context size, numerical precision, and the number of layers and heads. Reusing a block adds computational steps while keeping its local architecture fixed. Can sufficiently many recurrent rounds compensate for low precision, or does the input interface impose a limit that additional computation cannot overcome?

We study static-source recurrent attention, where mutable slots repeatedly query immutable, position-locally encoded inputs. For fixed query dimension and kernel count, disjointness requires $TM = \Omega(N)$ and $P = \Omega(\log N)$. The precision requirement persists under constant error, even with unrestricted recurrent time and mutable memory. Internal computation and input-access capacity are therefore distinct resources.

A finite response table captures all information an input block can supply through the interface. Conversely, floating-point repair supports exact machine simulation once precision reaches a logarithmic scale. Our central question is what information recurrent computation can acquire through a fixed input interface.

1.1 ARCHITECTURAL CORRESPONDENCE

In the original Perceiver's local-input configuration, input elements carry positional features and learned latents repeatedly cross-attend to them (Jaegle et al., 2021). Its access graph has the form

$$X_i = \phi(u_i, \pi_i), \qquad Z_0 = Z_{\text{learned}}, \qquad Z_{t+1} = F_t(Z_t, \text{CrossAttn}_{j_t}(Z_t, X)) .$$

Here X is immutable and F_t acts on latent state. Sharing source projections keeps the kernel library fixed as reads are repeated. The dot-product Read–Process encoder, also known as Set2Set, similarly queries elementwise embeddings with an evolving recurrent state (Vinyals et al., 2016). Figure 1 shows this common access structure.

Two nearby architectures expose different interfaces. Slot Attention normalizes across slots, so a slot's read depends on the entire query array (Locatello et al., 2020). With scalar keys and values

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

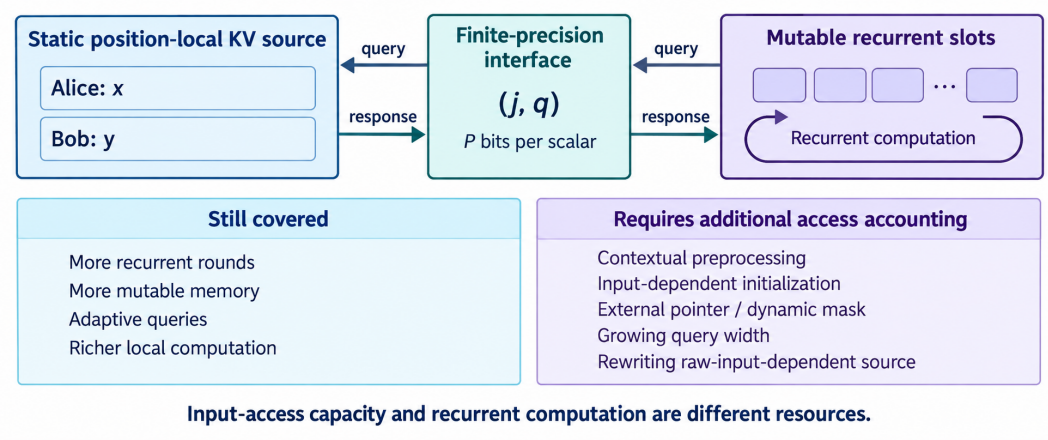


Figure 1: Input-access capacity versus internal computation. Queries and responses cross a finite interface; the recurrent loop acts on mutable slots. More rounds or storage do not enlarge the source-response profile. The lower band identifies changes that require explicit access accounting.

$(0, 1)$ and queries $(q_1, q_2) = (0, a)$, its idealized read is $r_1 = 2/(3 + e^a)$, omitting the stabilizing offset. Holding q_1 fixed therefore does not fix its response. BLIP-2's visual Q-Former instead reads fixed but contextual image features (Li et al., 2023); immutability alone does not supply position locality. Appendix G.1 derives the slot example and accounts for contextual preprocessing.

1.2 Related work

Communication and precision bounds. Sanford et al. (2023); Bhattamishra et al. (2024) use communication complexity to establish architectural limitations and separations. Merrill & Sabharwal (2023b) place log-precision Transformers within constant-depth logspace-uniform threshold circuits; Merrill & Sabharwal (2023a) express their classifiers in first-order logic with majority quantifiers. These results make precision an explicit computational resource. Our cumulative query bound follows the communication approach. The response-table bound captures all answers available through a fixed finite source interface, including the quantities needed to reconstruct rounded softmax. Its size is independent of recurrent time and mutable memory.

Universality and finite-precision simulation. Pérez et al. (2021) establish Turing completeness for hard-attention Transformers. Chung & Siegelmann (2021) develop bounded-precision recurrent simulations with packed stack storage and constant transition overhead, while Giannou et al. (2023) construct programmable looped Transformers. We adopt the packed storage organization and implement its addressed transfers through finite-precision softmax. Two-rounding lattice repair recovers transported values exactly before subsequent updates, supporting fixed rational weights, constant local dimensions, and $O(S + p)$ persistent mutable bits. The required precision depends on input and workspace size rather than execution length.

Intermediate computation and input access. Merrill & Sabharwal (2024) study how generated scratchpad length changes decoder expressivity. Their intermediate tokens enlarge the context available to later computation. We keep the source immutable and ask which input-access bottlenecks persist even with unrestricted recurrent computation. The distinction concerns the source interface: our mutable state can still grow and support general computation. Related constructions study low-precision softmax with ordinary or summarized chain-of-thought (Brösamle & Eckstein, 2026), and constant-bit simulation with a moving context window (Li & Wang, 2025). These models allocate access, context, width, and precision differently. Appendix J gives the detailed comparison.

2 MODEL AND RESOURCE ACCOUNTING

The input has n symbols. A position-local map converts each symbol and public metadata into keys and values without inspecting other symbols. These records are immutable. There are $s \geq 1$ mutable slots of width at most d. One round repeats a fixed sequence of L affine/ReLU and attention sublayers, with at most H heads per attention sublayer. Mutable-slot attention is bidirectional. Static-input attention uses a library of J kernels with fixed projections and fixed reduction semantics. Initial mutable contents are input-independent; halting is controlled by a stored flag.

For the lower bound, a scalar belongs to an alphabet $\mathbb{F}$ of size at most 2^P. Every cross-block numerical intermediate, including maxima and accumulators, has at most P bits. Local computation may be any deterministic operation. A query has d_k coordinates and its output has d_v coordinates. Any mutable mask, temperature, bias, or projection control must be included in the query description; an external uncharged address is unavailable. A fixed collection of input banks can be represented by a fixed kernel library.

Reference numerical semantics. Let ℓ_i be the rounded score at position i and let E be a deterministic finite-format exponential. A row evaluates

$$z = \max_i \ell_i, \qquad a_i = E(\mathrm{RN}(\ell_i - z)), \qquad D = \mathrm{fold}_{i=1}^{n} \mathrm{RN}(D + a_i), \tag{1}$$

$$w_i = \mathrm{RN}(a_i/D), \qquad y_r = \mathrm{fold}_{i=1}^{n} \mathrm{RN}(y_r + \mathrm{RN}(w_i v_{i,r})). \tag{2}$$

Both folds start at zero and follow input order. Invalid encodings receive fixed defaults, and executions must be numerically well-defined. We also cover prescribed binary reduction trees when the two input blocks form the two root subtrees. Arbitrary reductions require their own cross-cut accounting (Appendix A).

We distinguish recurrent rounds T, allocated persistent mutable bits $M = \Theta(sdP)$, total input-attention rows Q, and arithmetic work W. In particular, $Q \leq LHsT$. Static input records and their initialization are accounted for separately. For the compiler, mutable-slot metadata and scratch coordinates are included in M.

Randomized source interfaces. Let ξ be an input-independent seed containing all random tapes. A *seedwise-static* source call is determined by (ξ, j, q): conditional on ξ, at most J kernels have fixed records, masks, numerical operations, and reduction schedules. Controller updates and query selection may be randomized. Programs halt on every input and admitted seed; Q, T are worst-case bounds, and M is allocated storage. Appendix I.5 treats almost-sure halting and expected queries.

Proposition (Architectural correspondence)**.** *Suppose source records are immutable functions $\rho_j(u_i, \pi_i)$ of individual inputs and public metadata; initialization is input-independent or implemented by counted source calls; and all later input access is through $(j, q) \in [J] \times \mathbb{F}^{d_k}$. If source evaluations use the reduction semantics above with P-bit boundary scalars, the architecture inherits the communication bounds of Theorems 1 and 3, the latter under the seedwise-static condition, with its original source-row count Q.*

Appendix G.1 proves the proposition. Latent-only computation may vary between calls; the TM consequence additionally uses $Q \leq LHsT$.

3 A COMMUNICATION LAW WITH FINITE-INTERFACE SATURATION

Partition the input into contiguous blocks x and y held by Alice and Bob. All weights, metadata, initialization rules, and input lengths are public. Let $\mathrm{D}(f)$ denote deterministic two-party communication complexity.

Theorem 1 (Static-source communication law)**.** *If a deterministic static-source recurrent-attention program computes $f(x, y)$ using at most Q input-attention rows, then for an absolute constant C,*

$$\mathrm{D}(f) \leq C \min\Big\{(d_v + 1)PQ,\ J(d_v + 1)P\, 2^{(d_k+2)P}\Big\}. \tag{3}$$

The second term holds for arbitrary mutable memory and any execution that halts on every input. The bound holds for the ordered folds in Equations (1)–(2) *and for root-aligned binary reductions.*

The first term charges the numerical summaries exchanged for each query. The second enumerates all possible queries and global accumulator values in a one-way response table of size at most

$$B_{\text{off}} := JP\left(2^{d_k P} + 2^{(d_k+1)P} + d_v 2^{(d_k+2)P}\right). \tag{4}$$

This table answers every subsequent adaptive query. Appendix B gives both protocols and their proofs.

Corollary 2 (Disjointness resources). *Let* $\mathrm{DISJ}_N(x, y) = 1$ *if* $x_i y_i = 0$ *for every* $i \in [N]$. *If* d, H, L, J, d_k, d_v *are constants, exact computation on all inputs requires*

$$TM = \Omega(N), \qquad P = \Omega(\log N). \tag{5}$$

The precision bound holds even with unbounded mutable memory and halting time.

The precision bound constrains input access even with unrestricted mutable storage. Increasing query width d_k or kernel count J enlarges the response table, as quantified in Equation (3).

3.1 Bounded error and average-case precision

Write $\mathsf{R}^{\text{pub}}_{\varepsilon}(f)$ for public-coin communication complexity with worst-case error at most ε, and use $\to$ for one-way communication from Alice to Bob. Bob produces the answer.

Theorem 3 (Randomized finite-interface simulation). *A seedwise-static program computing* f *with worst-case error at most* ε *satisfies*

$$\mathsf{R}^{\text{pub}}_{\varepsilon}(f) \le \min\{C_0(d_v+1)PQ, B_{\text{off}}\}, \tag{6}$$
$$\mathsf{R}^{\to,\text{pub}}_{\varepsilon}(f) \le B_{\text{off}}.$$

For fixed dimensions and kernel count, any such program computing DISJ_N *with a fixed* $\varepsilon < 1/2$ *requires*

$$PQ = \Omega_{\varepsilon}(N), \qquad TM = \Omega_{\varepsilon}(N), \qquad P = \Omega_{\varepsilon}(\log N). \tag{7}$$

The offline bound and precision consequence require no uniform bound on recurrent time or mutable memory.

Appendix I.1 proves the simulation and applies randomized disjointness lower bounds (Kalyanasundaram & Schnitger, 1992; Razborov, 1992).

The one-way structure gives a separate average-case result. Let X be uniform on $\{0,1\}^N$, let I be independent and uniform on $[N]$, and set $Y = e_I$. Under this balanced distribution, denoted μ_{idx}, the answer is $1 - X_I$. Define binary entropy by $h_2(t) = -t\log_2 t - (1-t)\log_2(1-t)$, with $0\log_2 0 = 0$.

Theorem 4 (Average-case precision). *If a deterministic or randomized seedwise-static program has average error at most* $\varepsilon < 1/2$ *under* μ_{idx} *and its randomness, then*

$$B_{\text{off}} \ge N(1 - h_2(\varepsilon)). \tag{8}$$

Its success probability is at most

$$\min\left\{1, \frac{1}{2} + \sqrt{\frac{(\ln 2)B_{\text{off}}}{2N}}\right\}. \tag{9}$$

Thus $P = o(\log N)$ *gives success at most* $1/2 + N^{-1/2+o(1)}$ *for fixed* d_k, d_v, J.

Appendix I.2 proves the average-case bound by encoding the independent bits of X in Alice's response table. The one-way restriction is essential: with interaction, Bob can send I and receive X_I using $O(\log N)$ bits. Appendix I.3 analyzes resampled source masks and the resulting change in table capacity.

4 Exact regenerative reads and a fixed-weight compiler

We now use a specific floating-point member of the model. Let $\mathbb{F}_p$ have p fraction bits, normal exponents in $[-16p, 16p]$, gradual underflow, and round-to-nearest-even. A scalar occupies $P = p + O(\log(p+2))$ bits. Intermediate operations retain the displayed rounding nodes. With $u = 2^{-p}$, require

$$E_p(0) = 1, \qquad E_p(t) \ge 0, \qquad |E_p(t) - e^t| \le 2u \quad (t \le 0). \tag{10}$$

Appendix C provides a polynomial-time, $O(p)$-workspace implementation of this accuracy contract.

4.1 REPAIRING A SOFTMAX READ

Lemma 5 (Floating-point lattice repair). *Let $p \geq 3$, $r \in \{0, \ldots, p-3\}$, $\delta = 2^{-r}$, and $C_r = 2^{p-r}$. If $v \in [0, B] \cap \delta\mathbb{Z}$ is representable, $C_r > 2B$, and $z \in \mathbb{F}_p$ satisfies $|z - v| < \delta/4$, then*

$$\mathcal{Q}_{p,r}(z) = \mathrm{RN}_p(\mathrm{RN}_p(C_r + z) - C_r) = v. \tag{11}$$

The repair uses two separately rounded additions with coefficients 1 and -1; C_r is supplied as format metadata. With logarithmic precision and an addressed score gap, the softmax error lies within the repair radius. Appendix C proves the lemma and specifies the score construction and numerical parameters.

4.2 SPACE-PRESERVING MACHINE SIMULATION

Theorem 6 (Fixed rational-weight compiler). *Fix a deterministic multitape Turing decider A with read-only input bounded by end markers. Suppose it runs for at most τ steps on length-n inputs and visits at most S work cells, where the allocation bound S is public and uniform. There is a rational-weight recurrent-attention program depending only on A that uses*

$$\begin{aligned} p &= \Theta_A(\log(n + S + 2)), & s &= O_A(S/p + 1), \\ M &= O_A(S + p), & T &= O_A(\tau + 1), \end{aligned} \tag{12}$$

with $d, H, L = O_A(1)$, and returns the exact acceptance bit. Its coefficients are independent of n, S, p, τ. The guarantee uses the floating-point model above, uniform positional/format metadata, and internally controlled halting.

The compiler packs work-tape bits into addressed words and repairs every softmax read before updating the simulated state. Each machine transition uses a constant number of transfers, with exact canonical states at transition boundaries. Precision depends on the addressed size rather than the execution length. The storage construction and induction appear in Appendix D.

5 TASK CONSTRUCTIONS AND SEPARATION

Disjointness. For every $1 \leq k \leq N$, k workers can compare corresponding bits in batches and accumulate overlap flags. Repaired reads and a final Boolean aggregation give

$$P = \Theta(\log(N + 2)), \quad M = O(k \log(N + 2)), \quad T = O(\lceil N/k \rceil + 1). \tag{13}$$

The construction uses uniform length-dependent batch increments and aggregation coefficients (Appendix E). It matches the precision lower bound, with a logarithmic gap between the lower and upper bounds on TM.

Disjointness versus OR. At $P = \Theta(\log N)$ and $M = \Theta(\log N)$, OR on N bits takes one global attention read followed by a threshold, hence $O(1)$ rounds. Disjointness on two N-bit blocks requires $\Omega(N/\log N)$ rounds by Corollary 2. Thus equal precision and workspace budgets allow different recurrent depths for different tasks. Appendix E.1 proves the Boolean aggregation used by both constructions.

Parallel access. If at most a input rows can be evaluated per round, Theorem 1 gives $T = \Omega(N/(aP))$ for disjointness. At $M = \Theta(N \log N)$ and $P = \Theta(\log N)$, N workers attain constant rounds; a constant-row port requires $\Omega(N/\log N)$ rounds. These are round bounds: dense evaluation of the disjointness construction takes $O(N^2 + Nk)$ arithmetic work. Appendix F gives the sequential time and workspace bounds.

6 END-TO-END COMPILER EXECUTION

We instantiate the fixed-weight compiler as a seven-phase recurrent program over fixed-width slot arrays, with explicit finite-format arithmetic. A binary-palindrome decider and a binary-countdown tape-shuttle machine are checked against an independent explicit-tape interpreter after every complete transition. In 9 configurations satisfying the theorem's sufficient parameters, all 11,751 transitions agree exactly, including a 10,010-transition run. Across 338,795 attention rows, 9,780 raw responses are nonexact; repair restores every transported coordinate, with zero canonical-state and lattice deviation. At fixed $R = 128$ and $p = 352$, runs of 130, 1,040, 10,010 transitions all remain exact.

Removing repair or fusing its two rounding stages causes the first semantic divergence at transition 63 under the same parameters (Figure 2). We also test all 62 binary inputs of lengths 1–5 and sweep precision below the sufficient bound. Appendix H reports all 118 configurations, including failures, along with trace checks and resource counts. The exponential primitive uses high-precision evaluation followed by format rounding.

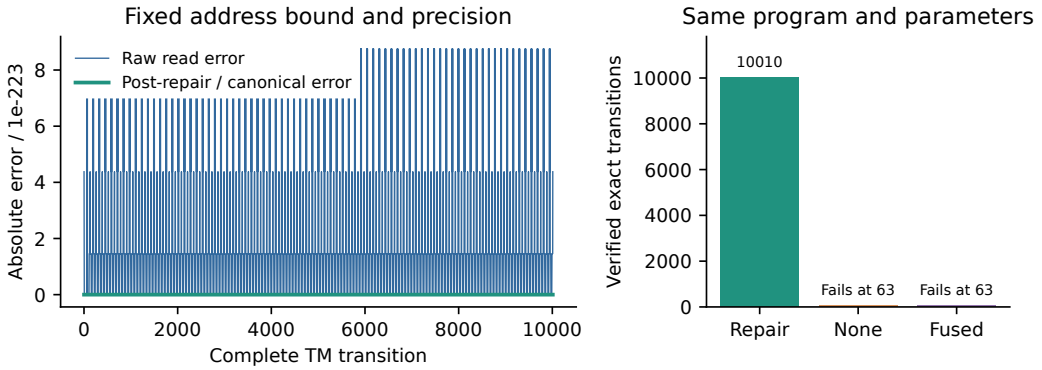


Figure 2: Approximate reads and exact recurrent state. Left: maximum raw read error within each transition of the 10,010-transition shuttle run; post-repair and canonical-state errors are exactly zero. A linear vertical scale displays zero directly. Right: identical parameters with repair removed or fused; both first diverge at transition 63.

AI USE STATEMENT

OpenAI Codex assisted with manuscript drafting, proof review and revision, supplementary verification and end-to-end compiler code, LaTeX preparation, bibliographic checks, architecture-figure generation, and execution of the verification scripts.

ETHICS STATEMENT

This work studies mathematical properties of a specified computational model. The reported checks use synthetic inputs and do not involve human participants or personal data.

REPRODUCIBILITY STATEMENT

Appendices A–G give the deterministic proofs and architecture mapping; Appendix I gives the randomized results. Appendix H specifies the end-to-end compiler experiments and component checks. The supplement provides both independent executors, fixed program definitions, exact-arithmetic code, all configurations and results, compressed read traces, figures, and reproduction commands. All test inputs are synthetic; generated inputs use seed 260919.

REFERENCES

Satwik Bhattamishra, Michael Hahn, Phil Blunsom, and Varun Kanade. Separations in the representational capabilities of transformers and recurrent architectures. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.09347.

Moritz Brösamle and Stephan Eckstein. The expressive power of low precision softmax transformers with (summarized) chain-of-thought. *arXiv preprint arXiv:2605.18079v2*, 2026. URL https://arxiv.org/abs/2605.18079v2. Accepted to ICML 2026.

Stephen Chung and Hava Siegelmann. Turing completeness of bounded-precision recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/ef452c63f81d0105dd4486f775adec81-Paper.pdf.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Mika Göös and Thomas Watson. Communication complexity of set-disjointness for all probabilities. *Theory of Computing*, 12(9):1–23, 2016. doi: 10.4086/toc.2016.v012a009. URL https://theoryofcomputing.org/articles/v012a009/.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. doi: 10.1137/0405044. URL https://doi.org/10.1137/0405044.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Qian Li and Yuyi Wang. Efficient turing machine simulation with transformers. *arXiv preprint arXiv:2512.00003v2*, 2025. URL https://arxiv.org/abs/2512.00003v2.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11525–11538, 2020. URL https://arxiv.org/abs/2006.15055.

William Merrill and Ashish Sabharwal. A logic for expressing log-precision transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://papers.neurips.cc/paper_files/paper/2023/hash/a48e5877c7bf86a513950ab23b360498-Abstract-Conference.html.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023b. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31/.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/1f59721c106ea80f613299039112f651-Abstract-Conference.html.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021. URL https://www.jmlr.org/papers/v22/20-302.html.

Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. doi: 10.1016/0304-3975(92)90260-M. URL https://doi.org/10.1016/0304-3975(92)90260-M.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.02896.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06391.

A THE NUMERICAL AND ACCESS MODEL

An input consists of n symbols. A fixed number of position-local maps turn a symbol and public positional metadata into keys and values. The maps may depend uniformly on the input length, allocated workspace, and numerical format, but cannot inspect other input symbols. Input records are never updated. An encoder that contextualizes the input using other positions is outside this definition unless its computation and communication are separately charged.

There are s mutable slots, each with at most d numerical coordinates. A round is a fixed sequence of L affine/ReLU and attention sublayers, with H heads per attention sublayer. We allow bidirectional attention among mutable slots and cross-attention from mutable slots to the entire static input. The layer sequence is reused each round. There is a library of J cross-attention kernel types, each having fixed key/value projections, reduction semantics, and any fixed mask. Any mutable control of a mask, bias, temperature, or value projection must be included in the query description; an external uncharged address is forbidden. In the basic model, cross-attention is unmasked. A fixed finite collection of separately named input banks is equivalent to a constant-size kernel library.

For the lower bounds a scalar belongs to an arbitrary finite alphabet $\mathbb{F}$ with $|\mathbb{F}| \le 2^P$. All intermediate scalars exposed across an input-block boundary, including scores, maxima, denominators, and partial output accumulators, occupy at most P bits. Extra guard bits must be included in P. Local operations may be arbitrary deterministic functions on their finite inputs. This strengthens the lower-bound model beyond the ReLU implementation needed for the upper bound.

A cross-attention row receives $q \in \mathbb{F}^{d_k}$. It computes scores from (q, k_i), finds their maximum z, evaluates nonnegative finite-precision exponentials a_i, accumulates a denominator D, and forms weights w_i and output coordinates. A particular reference evaluation is

$$z = \max_i \ell_i, \qquad a_i = E(\mathrm{RN}(\ell_i - z)), \qquad D = \mathrm{fold}_{i=1}^{n} \mathrm{RN}(D + a_i),$$

$$w_i = \mathrm{RN}(a_i/D), \qquad y_r = \mathrm{fold}_{i=1}^{n} \mathrm{RN}(y_r + \mathrm{RN}(w_i v_{i,r})), \quad 1 \le r \le d_v,$$

where folds begin at zero and follow input order. Invalid numerical encodings are assigned a fixed default; actual computations are required to be well-defined. Scores may include the usual attention scaling. The exact choice of exponent approximation is irrelevant to the communication theorem as long as it is a deterministic scalar operation. A kernel that accumulates an unnormalized numerator before dividing admits the same theorem with a slightly smaller table.

This serial definition specifies the numerical reference order. Section 5 accounts separately for the work required to evaluate a round. Balanced or fused reductions can also be studied, but their crosscut summaries and temporary workspace must be charged. In particular, floating-point addition must not be treated as associative.

The resource vector is

$$(T, M, P, Q, W; J, d_k, d_v, \mathcal{A}),$$

where T is the number of recurrent rounds, $M = \Theta(sdP)$ is allocated persistent mutable storage, Q is the total number of evaluated input-attention rows, W is actual arithmetic or bit work, and $\mathcal{A}$ fixes access and reduction structure. Initial mutable contents are input-independent. Halting is determined by a stored flag, not by a nonuniform input-dependent external clock. Static positional/format metadata are separately recorded; for the compiler we also charge their mutable-slot copies to M conservatively.

B A STATIC-SOURCE COMMUNICATION LAW

Partition the input into two contiguous blocks x and y, owned respectively by Alice and Bob. Public input length, metadata, weights, initialization, and architecture are known to both parties. Write $\mathrm{D}(f)$ for deterministic two-party communication complexity. The parties have unbounded local computation. The communication bound therefore permits arbitrary deterministic computation between input queries.

Theorem (restated) (Static-source communication and saturation)**.** *Suppose a deterministic recurrent-attention implementation of $f(x, y)$ evaluates at most Q static-input rows and uses a library of J kernels as defined above. There is an absolute constant C such that*

$$\mathrm{D}(f) \le C \min\left\{(d_v+1)PQ, J(d_v+1)P\,2^{(d_k+2)P}\right\}. \tag{S}$$

The second bound places no restriction on the number of mutable slots, local computation steps, or recurrent rounds, except that the implementation halts on all inputs. The kernel library is public and the initial mutable state is input-independent.

Proof. For the first bound, both parties maintain identical copies of the mutable state. Thus they know each next query, all mutable-to-mutable attention results, and all local updates. To evaluate one input row, Alice computes the maximum score over her block and transmits it to Bob, who completes the maximum and returns it. With the global maximum known, Alice computes the denominator fold over her block beginning at zero and sends the resulting P-bit accumulator. Bob continues the same fold over his block and returns the final denominator. Alice then computes each weighted-numerator fold over her block beginning at zero, sends its d_v accumulators, and Bob continues each fold and returns the outputs. This exchanges $O((d_v+1)P)$ bits. It reproduces the specified operation order exactly, including every rounding event. Repeating the procedure, and locally applying all state updates and the halt test, costs $O((d_v+1)PQ)$ bits. Data-dependent early halting is public once the state is replicated; it cannot encode uncharged extra information.

For the second bound Alice sends a single offline table before the recurrent simulation begins. Fix one kernel. For every $q \in \mathbb{F}^{d_k}$ she records the maximum of her scores, denoted $Z_x(q)$. For every $(q, z) \in \mathbb{F}^{d_k+1}$ she records the denominator accumulator after processing her block from zero, denoted $D_x(q, z)$. Finally, for every $(q, z, D) \in \mathbb{F}^{d_k+2}$ she records the vector of weighted-numerator accumulators after processing her block from zero, denoted $Y_x(q, z, D) \in \mathbb{F}^{d_v}$. These entries are functions solely of Alice's input and public information. Entries with invalid numerical arguments can be filled with a fixed default.

The table length for one kernel is at most

$$P\,2^{d_k P} + P\,2^{(d_k+1)P} + d_v P\,2^{(d_k+2)P} \le 3(d_v+1)P\,2^{(d_k+2)P}.$$

After receiving the tables for all J kernels, Bob can evaluate any future query without Alice. He combines $Z_x(q)$ with the maximum over his block, continues the denominator from $D_x(q, z)$, and continues the output folds from $Y_x(q, z, D)$. The resulting output is bit-for-bit identical to the full input evaluation. Because the tables cover *every* possible query and every relevant global numerical state, adaptive queries introduce no difficulty. Bob locally simulates all mutable computation and halting, regardless of its duration or memory consumption. If the communication convention requires both parties to know the final Boolean answer, Bob sends one final bit, absorbed in the displayed constant. The offline construction also proves the same second bound for one-way communication from Alice to Bob. This proves the second bound and hence their minimum. □

Root-aligned tree reductions. The same communication bound holds when the denominator and each output are evaluated by prescribed binary reduction trees whose root separates the two input blocks. To see this, define $D_x(q, z)$ and $Y_x(q, z, D)$ as the roots of Alice's subtrees, rather than the terminal values of ordered folds. Bob evaluates his corresponding subtree and performs the prescribed rounded root addition. This is exactly the reference tree, not an associative regrouping of a serial fold. The per-row protocol communicates the same scalar summaries, and the offline tables have exactly the same dimensions. This proves (S) for this common mathematical parallel-reduction order as well. Reproducing the tree on a small-space sequential machine can require an additional reduction stack; its workspace must be added to the evaluation cost.

The theorem is a lower-bound transfer principle from classical communication complexity. An independently established lower bound $\mathrm{D}(f) \ge L_f$ yields both a total query-bandwidth bound and a finite-interface capacity bound. The protocol represents attention aggregation by communicating partial numerical evaluations.

Corollary (restated) (Unconditional disjointness law)**.** *Let*

$$\mathrm{DISJ}_N(x, y) = 1 \quad \Longleftrightarrow \quad x_i y_i = 0 \text{ for every } i \in [N].$$

For static-source recurrent attention with $d, H, L, J, d_k, d_v = O(1)$ and $M = \Theta(sP)$, using the ordered fold or the root-aligned tree just specified, exact computation on all inputs implies

$$TM = \Omega(N), \qquad P = \Omega(\log N). \tag{D}$$

The precision requirement holds even with unboundedly many slots and unbounded halting time.

Proof. We include the deterministic communication lower bound to make the reduction self-contained. For each $x \in \{0,1\}^N$, the pair $(x, \overline{x})$ is a 1-input. For distinct x, x', at least one of $(x, \overline{x'})$ and $(x', \overline{x})$ is a 0-input: inspect a coordinate at which x and x' differ. A fixed transcript of a deterministic two-party protocol defines a combinatorial rectangle, as follows by induction on its messages: Alice's messages restrict only her input among pairs sharing the previous transcript, and Bob's messages restrict only his. A monochromatic 1-rectangle therefore contains at most one of these 2^N pairs. A protocol using at most c communicated bits has at most 2^c terminal transcripts after padding to a fixed maximum length. Thus $c \geq N$.

The first part of (S), together with $Q \leq LHsT = O(sT)$, gives $N \leq CPsT = O(TM)$. The second gives $N \leq CP2^{C'P}$ for fixed local dimensions and kernel count. Taking logarithms yields $\log N \leq O(P) + \log P$, and hence $P = \Omega(\log N)$ for all sufficiently large N. □

The precision bound holds with unrestricted memory because a fixed-dimensional finite-precision *static query interface* has only a finite library of distinguishable input interactions; arbitrarily long adaptive computation cannot create a new interface value. Input-local preprocessing can be arbitrary and nonuniform in length under the communication model. Preprocessing that mixes the two blocks requires separate communication accounting.

C Ordinary floating-point reads with exact regeneration

The complementary upper bound uses a specific member of the preceding numerical family. Let $\mathbb{F}_p$ be signed binary floating-point numbers with p fraction bits, round-to-nearest-even, and normal exponent range $[-16p, 16p]$, with gradual underflow. A word needs

$$P = p + O(\log(p + 2)) = \Theta(p)$$

bits, including sign and exponent. Intermediate operations follow the displayed arithmetic graph without reassociation across its rounding nodes. Overflow does not occur in the construction. The exponent primitive satisfies, for $t \leq 0$,

$$E_p(0) = 1, \qquad E_p(t) \geq 0, \qquad |E_p(t) - e^t| \leq 2u, \qquad u = 2^{-p}. \tag{E}$$

A faithfully rounded binary exponential satisfies a stronger accuracy condition on this range. Exact correct rounding of every transcendental value is not needed. A deterministic polynomial-time, $O(p)$-workspace implementation of (E) suffices for the machine simulation; its existence is detailed below.

Lemma (Two-rounding lattice repair)**.** *Let $p \geq 3$, $r \in \{0, \ldots, p-3\}$, $\delta = 2^{-r}$, and $C_r = 2^{p-r}$. Let $v \in [0, B] \cap \delta\mathbb{Z}$ be representable, with $C_r > 2B$. If $z \in \mathbb{F}_p$ satisfies $|z - v| < \delta/4$, then*

$$\mathcal{Q}_{p,r}(z) = \mathrm{RN}_p(\mathrm{RN}_p(C_r + z) - C_r) = v. \tag{R}$$

For $v = 0$ the strict $\delta/4$ radius also handles the finer spacing immediately below the binade boundary C_r.

Proof. In $[C_r, 2C_r)$ adjacent floating-point numbers are spaced by $C_r 2^{-p} = \delta$. Thus $C_r + v$ is a representable grid point. For $v > 0$, the condition $|z - v| < \delta/4$ places $C_r + z$ strictly inside its rounding cell. For $v = 0$, the closest point below C_r is at distance $\delta/2$, so the left rounding boundary is $C_r - \delta/4$; the same strict hypothesis applies. The upper bound on B keeps all values within the relevant binade. The first rounding therefore returns $C_r + v$ exactly. Its difference with C_r equals representable v, so the second rounding is exact. □

The operation uses weights 1 and -1, with C_r supplied as a static format coordinate. Two separately rounded affine stages implement it. ReLU acts as the identity on the positive first-stage coordinate. Residual-form networks can route this coordinate through dedicated scratch channels while preserving the two rounding nodes.

Lemma (Softmax read accuracy)**.** *Suppose m scores are exactly representable, their differences are evaluated exactly, one target j has score exceeding all others by at least γ, and scalar values lie in $[0, B]$. Assume that no intermediate operation overflows. Under (E) and the reference reduction, if $mu \le 1/32$, its output $\widehat{y}$ satisfies*

$$|\widehat{y} - v_j| \le 128mB(e^{-\gamma} + u) + 128mu. \tag{A}$$

In the construction $B \ge 1$, so the last term can be absorbed by doubling the constant. No wrong-position exponential is required to be zero.

Proof. Write $\theta = mu \le 1/32$ and let $\bar{a}_i$ be the computed exponentials. The target is $\bar{a}_j = 1$ exactly, and their off-target sum is at most $me^{-\gamma} + 2\theta$. Put $A = \sum_i \bar{a}_i \ge 1$. We use the elementary scalar bound

$$|\,\mathrm{RN}(z) - z| \le u|z| + \zeta, \qquad \zeta = 2^{-17p-1} \le u.$$

Under the no-overflow assumption, iterating this inequality for a nonnegative m-term sum gives

$$|\widehat{A} - A| \le 2\theta A + 2m\zeta.$$

Indeed, products of the relative-error factors are bounded by $(1+u)^m \le e^\theta \le 1 + 2\theta$, and the sum of the propagated absolute errors is at most $2m\zeta$. Applied to the denominator, this yields $|D - A| \le 4\theta A$, because $A \ge 1$ and $m\zeta \le \theta$. Hence $D \ge (7/8)A$ and $|A/D - 1| \le 5\theta$.

The computed weights $w_i = \mathrm{RN}(\bar{a}_i/D)$ satisfy

$$\sum_i |w_i - \bar{a}_i/D| \le uA/D + m\zeta \le 3\theta, \qquad \sum_i |w_i - \bar{a}_i/A| \le 8\theta.$$

In particular, $\sum_i w_i \le 1 + 8\theta$. Rounding the products $w_i v_i$ contributes at most $2\theta B + \theta$ total absolute error. Their unrounded sum is at most $(1 + 8\theta)B$, so the sum U of their rounded values is at most $2B + \theta$. Rounding the final nonnegative fold contributes at most

$$2\theta U + 2m\zeta \le 4\theta B + 3\theta.$$

Finally, $\sum_i \bar{a}_i v_i/A$ differs from v_j by at most $B(me^{-\gamma} + 2\theta)$. Combining this with the weight, product, and fold errors gives

$$|\widehat{y} - v_j| \le mBe^{-\gamma} + 16\theta B + 4\theta,$$

which is stronger than (A). Every value channel transported by the construction is nonnegative. □

Choose a public bound $R \ge n + s + 10$ on every source count and address magnitude, let

$$p \ge 128 + 32\lceil \log_2 R \rceil, \quad b = \lfloor p/16 \rfloor, \quad r = 2b, \quad \lambda = 2^{\lceil \log_2 p \rceil}. \tag{P}$$

Increasing the constant in R to account for a fixed program has no asymptotic effect. For $m, B \le R$ and $\gamma \ge \lambda$, (A) is strictly less than $2^{-r}/16$. In fact $e^{-\lambda} \le 2^{-p}$, $r \le p/8$, and the error is at most $512R^2 2^{-p}$, which is less than $2^{-r}/16$ under (P). Integer channels use $r = 0$ and the same stronger bound. Lattice repair therefore yields exact reads. The construction has no dependence on the total number of future reads.

Fixed-parameter addresses. For a source slot at integer address i, supply the static key

$$k_i = (2\lambda i, 2\lambda i^2, 0, 0),$$

and use query $q_j = (2j, -1, 0, 0)$. With the ordinary $1/\sqrt{d_k} = 1/2$ scaling for $d_k = 4$, the score is

$$\ell_i = \lambda(2ji - i^2) = \lambda j^2 - \lambda(i - j)^2. \tag{K}$$

The target $i = j$ is unique and its nearest competitor has gap λ. A dynamic j^2 register is unnecessary because the first term is row-constant. All products and subtractions are exactly representable under

(P): their integer significands need $O(\log R + \log p)$ bits, well below p. Keys and coordinates have magnitudes at most $O(pR^2)$ and remain within the exponent range. Input positions, workspace regions, and a fixed number of sentinel positions have disjoint public integer addresses. Sentinel and padding positions keep unused queries well-defined.

A broadcast head gives the unique controller logit λ and every other source logit zero, using a static controller-role feature. The same read bound and repair recover its bounded words and nonnegative integer descriptors. Both heads use unmasked attention.

An explicit exponent implementation. To meet (E), first compare t with $-32p$. Below this threshold, e^t is smaller than half the smallest positive subnormal of the chosen format, so returning zero agrees with ordinary round-to-nearest underflow. In the remaining range, write $a = -t \in [0, 32p]$ and approximate e^{-a} using the Taylor polynomial at zero with $K = O(p)$ terms. The Lagrange remainder is at most $a^{K+1}/(K+1)!$; taking, for example, $K \geq 1024(p+1)$ makes it exponentially smaller than 2^{-p}. Terms have magnitude at most $e^a = 2^{O(p)}$. Using Cp guard bits for a sufficiently large absolute constant bounds accumulated arithmetic and cancellation error below 2^{-p-3}. Each term is generated from the preceding term with multiplication and division by an integer, using $O(p)$-bit registers; no list of terms is stored. Clip the result to $[0, 1]$, round to $\mathbb{F}_p$, and special-case $t = 0$. This gives (E) in polynomial bit time and $O(p)$ workspace. Guard registers are internal workspace of the exponential primitive; its input and output remain in $\mathbb{F}_p$. This workspace is included in the sequential evaluation bound. A different standard implementation obeying (E) gives the same compiler proof. The verification code instead uses high-precision exponentiation followed by ordinary binary rounding.

D A FIXED-WEIGHT, CONSTANT-OVERHEAD COMPILER

Theorem (restated) (Regenerative space-preserving simulation)**.** *Fix a deterministic multitape Turing decider A with a read-only input tape bounded by end markers and a fixed finite work alphabet. Suppose its execution on length-n input takes at most τ steps and visits at most S work cells. The allocation bound S is a public, uniform function of the input length. There is a fixed rational-weight recurrent-attention program, depending only on A, such that for*

$$p = \Theta_A(\log(n + S + 2))$$

with a sufficiently large constant, it uses

$$s = O_A(S/p + 1), \quad d, H, L = O_A(1), \quad M = O_A(S + p), \quad T = O_A(\tau + 1). \tag{C}$$

It starts with blank work memory and local read-only input records, halts under internal control, and returns the exact acceptance bit. The guarantee uses ordinary round-to-nearest operations and the softmax accuracy contract (E); it does not require exact one-hot weights. All program coefficients are independent of n, S, p, τ. Uniform positional and format metadata do depend on these allocation and format parameters, and their size and initialization are charged below.

Proof. Replace the fixed work alphabet by fixed-length binary blocks, changing space and time by constants depending only on A. Each work tape is represented by a current bit and two stacks holding visited bits to its left and right, with nearest bits on top. Upon writing a and moving right, push a to the left stack and pop the right stack; an empty pop returns blank zero. A left move is symmetric. A stationary move changes only the current bit. The combined number of stored bits is at most a constant times the visited work span. The input remains on the read-only tape.

Encode a top-first word $a_1 \dots a_\ell$, $0 \leq \ell \leq b$, by

$$v(a_1 \dots a_\ell) = \sum_{j=1}^{\ell} (1 + 2a_j) 4^{-j}. \tag{B}$$

It lies on the $\delta = 4^{-b} = 2^{-2b}$ grid in $[0, 1)$. For nonempty valid words define $\mathrm{clip}(z) = \mathrm{ReLU}(z) - \mathrm{ReLU}(z - 1)$. Direct substitution gives

$$a_1 = \mathrm{clip}(4v - 2), \qquad \mathrm{push}_a(v) = \frac{1 + 2a + v}{4}, \qquad \mathrm{pop}(v) = 4v - (1 + 2a_1). \tag{L}$$

Indeed, if the first bit is zero then $v < 1/2$, and if it is one then $v \geq 3/4$. This gives a constant gap for the top-bit test. Push and pop are affine after this test and use only fixed rational coefficients.

Each stack has one active word in the controller and an array of full words in mutable slots. Let k be the number of full words below the active word. Its occupancy ℓ is encoded by two geometric markers

$$u_\ell = 4^{-\ell}, \qquad w_\ell = 4^{\ell - b}. \tag{G}$$

Empty and full are exactly $\mathrm{clip}(2u_\ell - 1)$ and $\mathrm{clip}(2w_\ell - 1)$. Pushing multiplies u by $1/4$ and w by 4; popping applies the reverse factors. Empty markers are $(1, \rho)$ and full markers are $(\rho, 1)$, where the bounded format constant $\rho = 4^{-b}$ is static metadata. Thus a reset uses no coefficient growing with b and no multiplication of a variable counter by a length-dependent scale.

Before a push into a full active word, write that word at the next array address, increment k, and reset the active word and markers to empty. Then execute one local push. Before a pop from an empty active word with $k > 0$, read the word at depth k, decrement k, reset the markers to full, and execute one local pop. If $k = 0$ and the active word is empty, return the blank bit. Each operation needs at most one array transfer; the bound is worst-case constant, not amortized. All inactive full-word slots remain canonical dyadic words, including stale values beyond the current depth.

Reads use (K), softmax accuracy (A), and repair (R). Array regions are disjoint; each of the fixed number of stacks has its own head and public base address. The target is the base plus k or $k + 1$, so only affine address calculations are needed. Input reads use the same construction at the read-only input head address. A broadcast head copies a write address and a new bounded word from the controller. At each slot, integer equality is computed by

$$E(z) = \mathrm{ReLU}\big(1 - \mathrm{ReLU}(z) - \mathrm{ReLU}(-z)\big), \qquad z = \text{local address} - \text{write address}. \tag{W}$$

For integer z, this is one exactly when $z = 0$. Address differences are exactly representable. The matching slot writes; other slots retain their words.

Finite control is implemented by affine/ReLU circuits as follows. For a bounded nonnegative candidate $v \in [0, B_0]$ and a Boolean e, with fixed rational B_0, the ReLU expression $\mathrm{ReLU}(v + B_0 e - B_0)$ equals ev. Clipped candidate formulas are therefore selected by a constant-size ReLU circuit. Signed candidates are split into positive and negative parts. Boolean conjunction, disjunction, and negation use fixed affine/ReLU formulas on $\{0, 1\}$. Machine state and microphase are represented by a constant-size one-hot vector. Each transition is a conjunction of a state indicator and the finitely many currently read symbol bits, followed by fixed Boolean combinations. Word candidates, markers, and bit registers are bounded by constants. Large integer addresses are never conditionally multiplied: their updates are $k \leftarrow k + e_+ - e_-$ and $j \leftarrow j + d$, where the increments are small exact integers determined by finite control. These updates use fixed coefficients.

A constant number of scratch channels route reads, repair their bounded or integer coordinates, evaluate the transition, and broadcast writes. Scratch channels can be cleared by a fixed negative identity in residual form. Metadata channels have zero updates. The finite microprogram has a constant number of phases: input read, transition selection, conditional spill/load, local push/pop, and final state update, with a fixed constant expansion into attention and feed-forward stages. A phase indicator disables irrelevant bounded assignments and sets address increments to zero. Queries need not be conditionally gated by a large-address multiplier. They may always evaluate base-plus-depth or base-plus-depth-plus-one. A sentinel at depth zero and one padding position beyond each array keep these expressions valid even in inactive phases; phase gates ignore the bounded returned words. Noncontroller input-head registers remain at a valid end-marker address. The layers implementing this finite program are repeated without changing their coefficients. Since A has a fixed number of tapes and states, all local dimensions and the number of sublayers are constants depending only on A.

Under (P), every transported word or integer is restored exactly before its use. Subsequent local operations are dyadic, use at most $2b + O(1)$ significant bits for words and $O(\log(n + S))$ for addresses, and are exactly representable. By induction on microphases, all stored machine registers and active words are canonical when the next machine transition begins. By induction on machine steps, the simulated configuration equals that of A. No error term proportional to τ arises. A stored halt flag ends the computation after the corresponding machine halt.

Each array has $O(S/b+1)$ cells and the controller has constant size. Each slot, including conservatively charged metadata channels and scratch registers, uses $O(p)$ bits. Hence $s = O(S/p+1)$ and $M = O(S+p)$. Each machine step uses a fixed number of recurrent microsteps, giving $T = O(\tau+1)$. For a fixed allocation and format, even a longer halting execution obeys the same exact invariant: precision depends on the addressed size, not on a promised time horizon. Choosing p as a sufficiently large constant times $\log(n+S+2)$ makes (P) hold because the allocated slot count is $O_A(S+1)$. □

The program coefficients are independent of input length. Its public metadata include λi, λi^2, $C_0 = 2^p$, and $C_r = 2^{p-r}$, generated from the allocation and format. Appendix G discusses the resulting architectural restrictions.

E Matching task constructions and access separation

E.1 Boolean aggregation with tied maxima

The final disjointness step aggregates several overlap flags, so its largest score need not be unique. The following argument treats that case directly.

Lemma (Boolean aggregation)**.** *Let $b_1, \dots, b_m \in \{0,1\}$, with scores $\ell_i = \lambda b_i$ and values $v_i = b_i$. Use the reference reduction and the parameters in (P), with $m \le R$. If all flags are zero, the output $\widehat{y}$ is zero. Otherwise,*

$$3/4 < \widehat{y} < 5/4, \qquad \mathrm{clip}(4\widehat{y}-2) = 1. \tag{OR}$$

The threshold is evaluated with the same rounded affine/ReLU operations.

Proof. Suppose $h \ge 1$ flags are one. Their computed exponentials are exactly one. Each other exponential is at most $e^{-\lambda} + 2u$. Write A for the sum of these computed exponentials in exact arithmetic. The corresponding normalized average satisfies

$$\frac{h}{A} \ge 1 - m(e^{-\lambda} + 2u), \qquad \frac{h}{A} \le 1.$$

The denominator, weight, product, and output-fold estimates in the proof of (A) require only nonnegative terms and $A \ge 1$. With values in $[0,1]$, they give

$$|\widehat{y} - h/A| \le 24mu.$$

Thus $\widehat{y} \ge 1 - me^{-\lambda} - 26mu > 3/4$, and $\widehat{y} \le 1 + 24mu < 5/4$ under (P). On this interval, multiplication by four and subtraction of two and three are exact in the binary format. The two ReLU terms in $\mathrm{clip}(4\widehat{y}-2)$ therefore differ by exactly one. When $h = 0$, every weighted value is zero and the threshold also returns zero. □

E.2 Parallel disjointness construction

Proposition (restated) (Disjointness construction)**.** *For every $1 \le k \le N$, a static-source recurrent-attention family computes DISJ_N with*

$$P = \Theta(\log(N+2)), \qquad M = O(k\log(N+2)), \qquad T = O(\lceil N/k \rceil + 1). \tag{U}$$

Thus $TM = O(N\log N)$ throughout $1 \le k \le N$, since $k(\lceil N/k \rceil + 1) \le N + 2k \le 3N$, while (D) gives $TM = \Omega(N)$. The precision threshold $\Theta(\log N)$ is tight.

Proof. Use k worker slots and one controller. Pad both input blocks publicly with zero symbols to a multiple of k, increasing their lengths by less than k. Worker r processes indices $r, r+k, r+2k, \dots$. Two address reads obtain the corresponding bits from x and y. Lattice repair yields exact Boolean inputs; $\mathrm{ReLU}(x_i + y_i - 1)$ computes their conjunction. Each worker accumulates its local disjunction using a fixed Boolean ReLU circuit. After $\lceil N/k \rceil$ batches, a controller obtains the disjunction of the worker flags. Give each worker score λb_i and value b_i, where b_i is its overlap flag. For this construction, take a public bound $R = O(N)$ covering the padded input, workers, and addresses, and choose p and λ by (P). The Boolean aggregation lemma gives the exact OR of the flags, including when several flags are one. The controller complements this bit to obtain DISJ_N.

All indices, flags, counters, and positional metadata use $O(\log N)$ bits per constant-width slot. The batch counter is stored and tested internally; the public batch size and endpoint may be supplied as uniform constants. This task construction permits uniformly generated length-dependent coefficients for batch increments and final aggregation; the fixed-weight assertion of the preceding compiler is a separate, stronger guarantee for sequential machine simulation. Rounding is controlled by the same conservative choice of precision and logit gap as before. The number of attention/feed-forward phases per batch is constant, proving (U). □

If only a cross-input rows are permitted per round, rather than one per worker, (S) gives

$$T \geq \Omega\left(\frac{N}{aP}\right). \tag{BAND}$$

For example, at the same memory allocation $M = \Theta(N \log N)$ and precision $P = \Theta(\log N)$, the $k = N$ construction has constant recurrent depth, whereas a constant-row input port requires $\Omega(N/\log N)$ rounds. Arbitrarily powerful local processing and passive extra memory do not eliminate this port restriction. The available number of input rows therefore affects the round bound even at a fixed memory allocation.

There is also a strict precision contrast within this fixed access regime. At precision $o(\log N)$, no exact family computes disjointness for all sufficiently large N, regardless of its mutable-slot budget or halting time. At precision $\Theta(\log N)$, the preceding construction computes it.

F Work accounting and the sequential simulation boundary

Proposition (restated) (Bit-machine evaluation)**.** *Consider the algorithmic submodel with constant local dimensions, a prescribed streamable numerical reduction, and a fixed uniform program for local operations and record generation. Put $B = P + \lceil \log_2(n + s + 2) \rceil$. Require each local primitive and each position-local input record, including generated metadata, to be computable in $\mathrm{poly}(B)$ time and $O(B)$ workspace. A T-round, s-slot computation can then be evaluated by a deterministic multitape bit machine in*

$$O\big(T(ns + s^2 + s)\,\mathrm{poly}(B) + I\big) \text{ time}, \qquad O(sP + P + \log(n + 2)) \text{ workspace}, \tag{REV}$$

where I is the charged initialization time. In the compiler regime $P = \Omega(\log(n + s + 2))$, the polynomial factor is simply $\mathrm{poly}(P)$. This proposition does not apply to the arbitrary, potentially inefficient local maps permitted by the communication lower bound. If halting is imposed by a public counter rather than a state flag, its $\log T$ bits must also be included.

Proof. Store current and next mutable arrays and a constant number of row accumulators. Generate each local input record on demand using its symbol and public position. For each cross-input row, scan the input a constant number of times for the maximum, denominator, and weighted numerator. Recompute exponents rather than storing an input-length vector of weights. Each row needs $O(n\,\mathrm{poly}(B))$ work and $O(B)$ scratch space. For each mutable-attention row, scan the current mutable array instead, using $O(s\,\mathrm{poly}(B))$ work. Constant-width local sublayers take $O(s\,\mathrm{poly}(B))$ work. The term $\log(s+2)$ in temporary index storage is absorbed by $O(sP)$. Repeat for the constant number of layers and T rounds, swapping old and new arrays after each sublayer. This preserves the reference rounding order exactly and proves (REV). □

For the compiler, the dense runtime arithmetic count is

$$W_{\mathrm{run}} = O_A\big((\tau + 1)(ns + s^2 + s)\big),$$

with scalar exponential calls counted as primitives; its bit cost multiplies this by $\mathrm{poly}(p)$. Immutable input records, if materialized, occupy $O_A(np)$ bits in addition to the n input symbols. Mutable-slot metadata occupy $O_A(sp)$ bits and were already charged to the conservative M bound. Local metadata and blank-state initialization take $O_A((n + s)\,\mathrm{poly}(p))$ bit work and $O_A(p + \log n)$ temporary workspace when streamed. The fixed rational program has $O_A(1)$ coefficients and a constant bit description; the uniform generator for metadata is a separate fixed program. When S is polynomially bounded in n, metadata generation is logarithmic-space. Input records and row weights are recomputed during each scan.

This distinguishes *persistent memory*, *temporary arithmetic workspace*, and the memory of a particular highly parallel hardware schedule. Simultaneously materializing all attention scores would consume additional temporary memory; it is not the schedule used to establish the workspace bound. For the disjointness construction, the dense work is $O(N^2+Nk)$, rather than $O(N/k)$: more worker slots reduce recurrent rounds, not proportionally the total arithmetic work.

OR illustrates why recurrent rounds and sequential work have different costs. A single input-attention row can compute OR using bit-dependent logits separated by $\Theta(\log n)$, followed by a threshold, with constant mutable width and $O(\log n)$ precision. It nevertheless evaluates n scores. A deterministic bit-query algorithm for OR must inspect all n bits on the zero input, since otherwise flipping an unqueried bit changes the answer without changing its execution. Hence no universal reverse inclusion can replace the n-dependent work in (REV) by only a polylogarithmic factor of the number of rounds. It also provides a second natural task family: at the same $M = O(\log n)$ and $P = \Theta(\log n)$ budgets, OR needs only constant rounds, whereas (D) forces $\Omega(n/\log n)$ rounds for disjointness on two equal input blocks.

G Architectural scope

G.1 Architecture mapping and contextual preprocessing

Proof of the architectural correspondence proposition. Collect every mutable tensor and control variable into the simulator's local state. Each input interaction is a counted source row. Position locality lets Alice construct her records from x and public metadata alone, while Bob constructs his from y. The online protocol reproduces each row through its boundary scalars; the offline protocol tabulates Alice's partial evaluations for every (j, q, z, D). Induction over local updates and source calls reproduces the original state and output with the same Q. Input-dependent initialization is simulated by its counted calls. Conditional on a seed, the same argument applies to the fixed source library and preserves the realized answer, hence the error probability after averaging. Latent-only layers require no communication even when their weights differ across steps. □

The correspondence concerns the access graph instantiated with the declared numerical reductions. Source-local normalization preserves locality, and normalization or self-attention on mutable state is local computation in the protocol. The compiler's repair construction imposes separate numerical requirements.

Collective queries in Slot Attention. For source keys k_i, values v_i, and slot queries q_j, the idealized competitive read is

$$a_{ij}(Q) = \frac{e^{k_i^\top q_j}}{\sum_\ell e^{k_i^\top q_\ell}}, \qquad r_j = \frac{\sum_i a_{ij}(Q) v_i}{\sum_i a_{ij}(Q)}.$$

Set $k_1 = v_1 = 0$, $k_2 = v_2 = 1$, $q_1 = 0$, and $q_2 = a$. Then $a_{11} = 1/2$, $a_{21} = 1/(1+e^a)$, and $r_1 = 2/(3+e^a)$. Adding the positive offset ϵ_0 used before the weighted mean (Locatello et al., 2020) gives

$$r_1 = \frac{(1+e^a)^{-1} + \epsilon_0}{1/2 + (1+e^a)^{-1} + 2\epsilon_0},$$

which still varies with a. A joint-query profile must therefore count all sd_k query coordinates and the numerical states of this reduction. It is not the fixed-dimensional per-slot profile in Theorem 1.

Charging contextual preprocessing. Suppose a preprocessing protocol communicates at most C_{pre} bits and produces a shared transcript. Conditional on that transcript and public randomness, each party can construct its final source records locally, and both know the reader's initial state and fixed source bank. Running preprocessing before the response-table protocol gives

$$\mathsf{R}^{\mathrm{pub}}_{\varepsilon}(f) \le C_{\mathrm{pre}} + B_{\mathrm{off}}.$$

Alice's table is now conditional on the transcript. The combined protocol can be interactive, so this extension alone does not give the one-way average-case bound. If $C_{\mathrm{pre}} = o(N)$, constant-error disjointness still requires $B_{\mathrm{off}} = \Omega(N)$. This identifies the extra quantity needed for frozen contextual features such as those read by the visual Q-Former. No value of C_{pre} is assumed for BLIP-2 here.

G.2 Numerical and architectural assumptions

The response tables require Alice's partial evaluations to depend only on her input and public parameters. Position-local encoding provides this locality; a contextual encoder must account separately for information exchanged across the partition. Masks, read pointers, and accumulator widths contribute to the finite interface. Fixed nonuniform weights remain public parameters, while query width and kernel count enter the table-size bound explicitly.

Local state normalization and softmax normalization play different roles. The lower bound allows arbitrary local normalization because the parties can compute it after reconstructing each query response. The fixed-weight compiler instead relies on preserved coordinates and separate rounding stages. The construction uses polynomial positional coordinates, logarithmic score gaps, and bidirectional mutable-slot attention. LayerNorm or RMSNorm would require an encoding compatible with those operations. The usual scaled dot product is included through a square head dimension. Fused kernels must preserve the rounding nodes used in the repair operation to inherit its guarantee.

For general fixed-point arithmetic, the communication and saturation laws remain valid whenever every cross-block scalar has P bits. The particular two-rounding repair is a floating-binade construction and does not automatically give the same fixed-weight compiler in a fixed-point model.

The reference reduction matters even when all final outputs occupy few bits. A balanced floating-point sum can require additional cross-cut messages or a recursion stack to reproduce bit-for-bit. Its accounting may therefore contain extra logarithmic factors. An implementation with wider accumulators must substitute that width into the communication theorem. The numerical kernel and its accumulator widths are thus part of the resource specification.

H End-to-end experiments and component verification

H.1 End-to-end programs and execution

The numerical executor stores its state in a fixed-width NumPy object array. Scalars are exact rationals rounded to $\mathbb{F}_p$; its row reductions preserve the specified operation order. High-precision exponentiation at $3p+128$ bits followed by rounding implements the exponential primitive, as in the component checks. Pure rounding and exponential evaluations are memoized; attention responses and machine transitions are not. This experiment instantiates a one-work-tape specialization of the compiler for two fixed machines.

The palindrome machine copies the binary input to its work tape, rewinds the input head, and compares the input against the reversed work-tape contents. The shuttle machine copies a binary counter, decrements it, and sweeps between the input end markers after each decrement. Its fixed transition table does not depend on the counter value or input length. Leading zeroes give different halting times at the same addressed size. For an n-bit counter of value v, it executes $(n+1)(2v+2)$ transitions.

Each transition repeats seven macro-phases: input read and repair; finite-control circuitry; transfer flags and broadcast; pointwise spill writes; addressed block loads and repair; local word/marker updates; and state commit with scratch reset. Each phase contains a fixed number of affine/ReLU stages for the chosen machine. The input read and block-load ports belong to the controller; all storage slots participate in dense read reductions and pointwise broadcast writes. Every broadcast recipient row is evaluated explicitly. Depth-zero sentinels and array padding support reads in inactive transfer phases. No work-memory lookup or write uses a data-dependent Python array index.

An independently written interpreter stores an explicit tape. At every transition boundary, the checker compares finite control, halt, input and work heads, exposed symbol, all logical tape contents through their packed words, stack depths, active occupancy markers, block counts and addresses, and the phase indicator. All stored words and markers, including stale slots, are tested for exact lattice membership. A mismatch raises an assertion and stops that configuration. Diagnostics identify target values only after numerical reads, and never feed those values into execution.

H.2 HORIZON, ABLATION, AND PRECISION RESULTS

The theorem-regime runs complete 11,751 transitions with 353 spills and 342 loads. All repaired responses equal their targets exactly; 3,762,181 off-target weights are nonzero. The fixed-size horizon sweep is:

R	p	Transitions	Attention rows	Spills	Loads
128	352	130	3,770	4	2
128	352	1,040	30,160	32	30
128	352	10,010	290,290	308	306

The no-repair condition feeds the raw softmax result into the same update circuit. The fused condition evaluates $\mathrm{RN}_p((C+z)-C)$ with one rounding node, so it returns the represented z. Both first diverge at transition 63. No other program parameter changes in these paired runs.

In a separate exploratory condition with $p = 32$ and the score gap reduced to $\lambda/4$, repaired execution first diverges at transition 7, while both ablations diverge at transition 1. These parameters are outside the sufficient regime.

The precision sweep uses $p \in \{8, 12, 16, 24, 32, 48, 64\}$ at three input/workspace sizes, testing one palindrome and one shuttle input at each point. It uses $b = \lfloor p/16 \rfloor$ for $p \geq 16$ and exploratory capacity $b = 1$ for $p = 8, 12$. Minimum successful tested precisions are summarized below; these describe only the tested inputs and horizons.

Input length	Common address bound	Minimum tested p
8	64	8
24	128	12
64	256	16

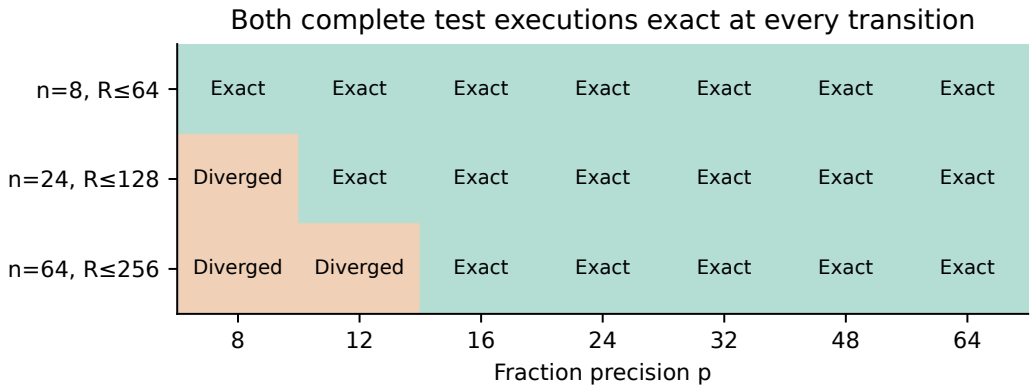


Figure 3: Exploratory precision sweep. Each cell records whether both complete executions match the reference at every transition. The common R bounds all tested allocations for that input length; actual slot counts vary with packed capacity. The theorem's sufficient precision is larger than these tested values.

Resource accounting and reproduction. The slot widths are 70 for the palindrome program and 72 for the shuttle program, including controller channels, scratch, metadata, and conservative padding in storage rows. For one work tape and capacity b, the allocation is $s = 1 + 2(\lceil S/b \rceil + 2)$. Each completed transition uses seven macro-phases, one raw-input row, two addressed mutable rows, and $2s$ broadcast rows. We record $P = p + 1 + \lceil \log_2(32p + 3) \rceil$ and charge $M = sdP$ bits. These counts have the stated $s = O(S/p + 1)$, $M = O(S + p)$ and $T = O(\tau + 1)$ scaling for $b = \lfloor p/16 \rfloor$. They are implementation counts, not a timing claim for dense attention.

The end-to-end runs used Python 3.10.12, mpmath 1.2.1, NumPy 1.21.5, and Matplotlib 3.5.1 on CPU. The supplement contains configuration JSON, every run result, compressed per-read traces, an implementation audit, and the original component scripts. Run from its root:

```
python3 code/make_configs.py
python3 code/run_suite.py --workers 4
python3 code/summarize_results.py
python3 code/write_report.py
```

H.3 Component verification summary

Table 1: Local verification results for the configurations in Appendix H.

Check	Cases	Result
Packed-word and occupancy identities	8,122	All passed
Floating-point lattice repair	7,283	All passed
Lattice boundary cases	5,744	All passed
Segmented-stack operations	19,844	All passed
Work-tape transitions	21,600	All passed
Scalar attention reads	144	All repaired exactly
Boolean aggregation, including ties	1,530	All passed
Disjointness fooling-set pairs	43,435	All passed
Ordered-fold response reconstruction	3,400	All matched
Root-aligned tree reconstruction	3,400	All matched

The accompanying programs use exact rational representations and explicit binary round-to-nearest-even emulation. The stack and lattice script checks 8,122 local coding/marker identities, 7,283 lattice-repair cases, 19,844 segmented-stack operations, and 21,600 simulated work-tape transitions against explicit reference objects. It checks 144 scalar attention reads using high-precision exponentiation followed by finite-format rounding. Across those cases, 690 off-target weights remain positive, and 114 raw read values differ from the target but are restored exactly. It also checks 43,435 disjointness fooling-set pairs. The smaller-gap settings are specified below.

A separate script checks 3,400 small two-block scalar attention computations against response-table reconstruction. It fills 1,677 distinct table entries lazily. In 646 cases, independently summing two denominator blocks and then adding them changes the result relative to the specified ordered fold; passing the running accumulator, as used in the proof, always matches. Another 3,400 cases verify the separately defined root-aligned binary-tree reduction and its subtree-response reconstruction.

The tests exercise local representation invariants and scalar kernels. Response-table entries are generated only for the queries encountered by the tests; the complete table-size bound follows from the enumeration argument in Appendix B.

H.4 Exact test configuration and execution

The local-code test enumerates binary words for capacities $b = 1, \dots, 9$ and all lengths up to each capacity. The lattice test uses $r = 0, \dots, 10$, $p = r + 24$, and perturbations $a\delta/32$ for $a = -3, \dots, 3$, together with integer-channel cases at $p \in \{24, 48, 80\}$. Segmented-stack and work-tape tests use pseudorandom seed 260918. Stack capacities range from 1 to 16, with 1,200 sampled operations at each capacity followed by draining; tape capacities range from 1 to 12 with 1,800 transitions each.

The attention test uses seed 9, source counts $m \in \{2, 5, 9\}$, every target position, and three value modes: a zero target, dyadic packed values, and integers. For each case it checks score-gap scales 1, $1/2$, and $1/8$ relative to the selected λ. Both smaller-gap settings are exploratory. The communication test enumerates all pairs of equal-length binary blocks with block length 1 through 4, fraction precision $p \in \{5, 8\}$, and scalar queries in $\{-3, -1, 0, 1, 3\}$. It tests the ordered fold and the separate root-aligned tree semantics on the same inputs.

The programs require Python 3.10 or newer and `mpmath`. The component-check results were reproduced with Python 3.14 and `mpmath` 1.4.1. From the supplemental package root, run:

```
python -m pip install -r requirements.txt
python code/verify_resource_laws.py \
  --out results/verification_results.json
python code/check_communication_profiles.py \
  --out results/communication_profile_results.json
```

Each script writes its counters and pass status to a JSON file.

H.5 Aggregation and repair boundary checks

The supplemental script `check_revision_edges.py` checks the Boolean aggregation argument and the lattice endpoints. It enumerates all Boolean flag vectors of lengths 1 through 8 at $p \in \{12, 16, 24\}$, using $\lambda = 2^{\lceil \log_2 p \rceil}$. All 1,530 cases return the exact OR after rounded thresholding; 1,398 have more than one flag equal to one, and 24 have only zero flags. These smaller formats satisfy $27m2^{-p} < 1/4$, the sufficient margin in the aggregation proof, although they are below the conservative global compiler setting (P).

A further 5,744 cases check lattice repair at $p = 3, \ldots, 16$ and $r = 0, \ldots, p-3$, including the zero target. The tested perturbations are $j\delta/32$ for $j \in \{-7, -4, -1, 0, 1, 4, 7\}$; the repair assertion is applied when the rounded input lies strictly within $\delta/4$ of the target. All cases pass. Run:

```
python code/check_revision_edges.py \
  --out results/revision_edge_results.json
```

I Randomized and distributional resource laws

I.1 Conditional simulation and worst-case error

The random seed ξ is independent of the inputs and includes the controller's random tape and any randomness used to initialize the source. The bound J holds uniformly over seeds. Conditional on ξ, a kernel's records and numerical operations are fixed, and repeated calls with the same (j, q) return the same response. The parties know the program and the seed.

Proof of Theorem 3. For the online protocol, the parties maintain identical copies of the mutable state and random-tape cursor. They execute each local operation using the specified tape and simulate each source call by the protocol in Appendix B. The source summaries cost at most $C_0(d_v+1)P$ bits per call. Induction over operations shows that the simulated state, stopping decision, and final answer equal those of the original execution on (x, y, ξ). The communication is at most $C_0(d_v+1)PQ$.

For the offline protocol, Alice constructs $Z^j_{x,\xi}(q)$, $D^j_{x,\xi}(q, z)$, and $Y^j_{x,\xi}(q, z, D)$ for each of the at most J kernels. These tables depend only on x, the public parameters, and ξ. Pad unused kernels and invalid arguments with fixed defaults. Their total length is at most B_{off}. Bob then simulates all local operations and source calls, continuing Alice's partial evaluations over his block. The simulated and original answers agree for every (x, y, ξ), so averaging over ξ preserves the error probability for every input. All communication precedes Bob's simulation, so neither the number of future calls nor the amount of mutable storage enters this bound. The root-aligned tree version uses subtree roots as before.

For any fixed $\varepsilon < 1/2$, randomized disjointness has public-coin communication complexity $\Omega_\varepsilon(N)$ (Kalyanasundaram & Schnitger, 1992; Razborov, 1992). Comparing this bound with each simulation gives $PQ = \Omega_\varepsilon(N)$ and $B_{\text{off}} = \Omega_\varepsilon(N)$ for fixed dimensions. Since $Q \le LHsT$ and $M = \Theta(sdP)$, the first inequality implies $TM = \Omega_\varepsilon(N)$. The second implies the precision bound. □

More explicitly, since $B_{\text{off}} \le 3J(d_v+1)P2^{(d_k+2)P}$, the precision constraint is

$$(d_k+2)P + \log_2 P + \log_2\big(J(d_v+1)\big) \ge \log_2 N - O_\varepsilon(1). \tag{14}$$

For fixed dimensions and J, considering separately $P \ge \log_2 N$ and $P < \log_2 N$ gives

$$P \ge \frac{\log_2 N - \log_2 \log_2 N - O_\varepsilon(1)}{d_k + 2}.$$

Small worst-case advantage. Suppose the success probability is at least $1/2 + \delta$ on every input. The all-probabilities bound of Göös & Watson (2016, Theorem 1.5), with acceptance parameters $\alpha = 1/2+\delta$ and $\beta = 1/2-\delta$, implies a communication lower bound $\Omega(\delta N)$ whenever δN exceeds an absolute constant. Combining the remaining constant-size regime with the offline bound yields

$$\delta \le \frac{C(B_{\text{off}} + 1)}{N} \tag{15}$$

for an absolute constant C. Thus fixed J, d_k, d_v and $P = o(\log N)$ imply worst-case advantage at most $N^{-1+o(1)}$.

I.2 A BALANCED DISTRIBUTION AND AN INFORMATION BOUND

Proof of Theorem 4. Let $V = V_\xi(X)$ be Alice's padded offline message, of length at most B_{off}. Given (V, ξ, i), Bob simulates the program on $Y = e_i$ and complements its answer to obtain an estimate $\widehat{X}_i$. All random tapes are included in ξ, so this estimate is deterministic conditional on (V, ξ, i). Put $\varepsilon_i = \Pr[\widehat{X}_i \neq X_i]$. Independence and uniformity give

$$H(X \mid \xi) = N, \qquad \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i \le \varepsilon.$$

Writing $E_i = X_i \oplus \widehat{X}_i$, the estimate and error determine X_i. Consequently,

$$H(X_i \mid V, \xi) = H(E_i \mid V, \xi) \le H(E_i) = h_2(\varepsilon_i).$$

Conditional subadditivity, concavity of h_2, and its monotonicity on $[0, 1/2]$ give

$$H(X \mid V, \xi) \le \sum_{i=1}^{N} h_2(\varepsilon_i) \le N h_2\left(\frac{1}{N}\sum_i \varepsilon_i\right) \le N h_2(\varepsilon).$$

On the other hand, $I(X; V \mid \xi) \le H(V \mid \xi) \le B_{\text{off}}$. Subtracting conditional entropies proves Equation (8).

If success is at most $1/2$, Equation (9) is immediate. Otherwise write success as $1/2 + \delta$ and apply the bound just proved with error $1/2 - \delta$. The binary entropy inequality

$$1 - h_2(1/2 - \delta) \ge \frac{2\delta^2}{\ln 2}$$

follows by differentiating twice: the left-hand side vanishes with zero derivative at $\delta = 0$ and has second derivative at least $4/\ln 2$. This proves Equation (9). Finally, $B_{\text{off}} = N^{o(1)}$ when $P = o(\log N)$ and the dimensions and kernel count are fixed. □

The balanced labels in μ_{idx} matter: under independent uniform $X, Y \in \{0,1\}^N$, the constant answer "not disjoint" already has error $(3/4)^N$.

Both simulations also preserve average error under any input distribution. Averaging over the seed yields a deterministic protocol with no larger average error.

I.3 FRESH SOURCE RANDOMNESS AND THE SIZE OF THE LIBRARY

Consider an enlarged source interface that, on every call, draws an independent uniform $I_t \in [N]$ and masks attention to positions x_{I_t} and y_{I_t}. Give the two positions equal scores and use their bits as values. The maximum, denominator, and output are exactly representable at constant precision, and the response is

$$a_t = \frac{x_{I_t} + y_{I_t}}{2} \in \{0, 1/2, 1\}.$$

The controller stores whether it has ever observed $a_t = 1$. After m calls it declares disjointness precisely when no such response occurred. If $h = \sum_i x_i y_i$, its false-negative probability for detecting an intersection is

$$\Pr[\text{no overlap detected}] = (1 - h/N)^m \le e^{-m/N} \quad (h \ge 1). \tag{16}$$

For $m = \lceil N \ln(1/\varepsilon) \rceil$, this is at most ε; disjoint inputs are always classified correctly. An internal binary counter uses $O(\log(m+1))$ constant-precision words and finitely many local steps per update.

The uncharged random position is an additional access resource. After fixing the seed, the source call depends on the invocation number, not just on (j, q). Across the execution there can be N distinct masks. For a library with at most K possible source realizations per nominal kernel, enumerate the realizations as separate kernels. The offline bound becomes KB_{off}, and Equation (14) becomes

$$(d_k + 2)P + \log_2 P + \log_2\big(JK(d_v + 1)\big) \geq \log_2 N - O_\varepsilon(1).$$

A mask sampled once and held fixed has $K = 1$ conditional on the seed, however many masks were possible before sampling.

Fresh stochastic rounding presents a related issue for the proof. Fixing a random tape can make the partial output depend on its invocation or cursor, giving a table of the form $Y_{x,\xi}(q, z, D, t)$. Reusing one sampled response for all equal arguments changes the joint law of the execution. The static-table argument therefore requires either a fixed rounding realization or an explicit bound on the additional modes. Independent stochastic rounding requires a separate precision analysis. The online protocol can still apply when the tape cursor is public and each realized call has bounded cross-cut summaries.

I.4 Private randomness and numerical output criteria

Private random tapes. If a program uses separate private tapes, the communication simulator may sample them in advance as parts of its public seed. Each simulated component follows its original transition rule and ignores tape bits it could not originally access. The execution distribution is unchanged. The source must still be static conditional on these tapes. If only the controller is randomized, Alice's offline table is independent of the controller tape, and Bob may generate that tape privately after receiving the table.

Represented outputs and margins. The simulations reconstruct the actual represented output, so any deterministic local decision rule can be applied exactly. For example, if a represented score g and threshold θ satisfy

$$(2f(x, y) - 1)(g - \theta) \geq \gamma > 0,$$

the protocol recovers the label by exact comparison. If the margin instead concerns an ideal score g^* and ideal threshold θ^*, bounds $|g - g^*| \leq \rho$ and $|\theta - \theta^*| \leq \zeta$ transfer it when $\rho + \zeta < \gamma$. A margin event holding with probability at least $1 - \varepsilon$ gives the corresponding bounded-error classification guarantee.

An expected margin requires a range bound. If $g - \theta \in [-B, B]$ for $B > 0$ and

$$\mathbb{E}\big[(2f(x, y) - 1)(g - \theta)\big] \geq \gamma$$

on every input, Bob can output a Bernoulli bit with parameter $1/2 + (g - \theta)/(2B)$. Its success is at least $1/2 + \gamma/(2B)$. Thus a constant normalized expected margin γ/B implies logarithmic precision under the same source assumptions. Similarly, for $g \in [0, 1]$, randomized rounding with parameter g has error exactly $\mathbb{E}|g - f|$.

An unnormalized expected margin alone is weaker. For example, sample one uniform index I once, use its fixed two-position mask to recover $b = x_I y_I$, and form the multiword score $g = 1/2 - Nb$. Then $\mathbb{E}g = 1/2 - h$, where $h = \sum_i x_i y_i$. Threshold zero has expected signed margin at least $1/2$ on every input, although the source precision is constant. The score range grows with N, so its normalized margin is only $\Theta(1/N)$. The score uses $O(\log(N + 1))$ constant-precision words.

I.5 Almost-sure halting and expected queries

Fix an input length, hence a finite input domain. If every input halts almost surely, then for every $\eta > 0$ there is a finite local-step cutoff whose timeout probability is at most η on every input. This follows by taking the maximum of finitely many per-input cutoffs. Applying the cutoff and returning a default bit increases worst-case error by at most η. The offline message remains of length B_{off}. Choosing fixed $\eta < 1/2 - \varepsilon$ therefore preserves the logarithmic precision conclusion without a deterministic runtime bound.

For expected query complexity let $\overline{Q} = \sup_{x,y} \mathbb{E}Q(x, y, \xi)$. Truncating after $\lceil \overline{Q}/\eta \rceil$ calls incurs probability at most η by Markov's inequality. If necessary, add a local-step cutoff with timeout probability at most η to handle long computations between calls. For fixed $\varepsilon + 2\eta < 1/2$, the online simulation then gives

$$P\big(\overline{Q} + 1\big) = \Omega_\varepsilon(N).$$

This implies $P\overline{Q} = \Omega_\varepsilon(N)$ for disjointness: consider an input of each label, and let p_0 be the probability of making no source call. Before the first call the computation is input-independent. Its errors on the two inputs sum to at least p_0, so $p_0 \leq 2\varepsilon$ and $\overline{Q} \geq 1 - 2\varepsilon$. The additive constant can consequently be absorbed. For fixed allocated storage and $Q \leq LHsT$ pathwise, the same argument yields $M \sup_{x,y} \mathbb{E}T = \Omega_\varepsilon(N)$.

J Extended related work

Communication and architectural limitations. Sanford et al. (2023) use communication arguments to analyze the representational strengths and limitations of attention. Bhattamishra et al. (2024) establish separations between Transformers and recurrent architectures. Our per-query simulation follows this general methodology. The additional result is an offline response-table bound for an immutable source, including the global numerical quantities required to reconstruct a rounded softmax evaluation. Its cost is independent of the number of adaptive queries. For bounded error, we use the classical randomized disjointness bounds of Kalyanasundaram & Schnitger (1992); Razborov (1992) and the small-advantage refinement of Göös & Watson (2016). The randomized extension conditions on the seed; the distributional result uses a one-way encoding of independent bits.

Precision, circuits, and logic. Merrill & Sabharwal (2023b) show that log-precision Transformers, under their feedforward computability assumptions, can be simulated by constant-depth logspace-uniform threshold circuits. Merrill & Sabharwal (2023a) give a first-order logical description with majority quantifiers for log-precision Transformer classifiers. These analyses constrain computation through circuit and logical expressivity. Our response-table argument instead bounds the input information available through an immutable finite interface. It permits unrestricted recurrent time and mutable storage, so its precision obstruction does not follow from a constant-depth computation bound.

Chain-of-thought computation. Merrill & Sabharwal (2024) relate the number of generated intermediate tokens to decoder expressivity. Under their generalized pre-normalization assumptions, polynomially many decoding steps characterize polynomial-time computation. Generated tokens extend the accessible context. Our controller can perform arbitrarily many recurrent updates, but its access to the original input remains mediated by a fixed source-kernel library and finite queries. The comparison concerns growing-context computation versus repeated access through a static interface.

Finite-precision recurrent simulation. Pérez et al. (2021) establish hard-attention Transformer universality, providing a foundational attention-based Turing-machine simulation. Our construction addresses finite-format softmax transport and persistent-space accounting. Chung & Siegelmann (2021) develop bounded-precision recurrent simulations with packed stack storage and constant transition overhead. We use their storage organization and implement its addressed transfers through finite-precision softmax. Floating-point lattice repair makes these transfers exact while retaining constant local dimensions and fixed rational coefficients.

Programmable and low-precision Transformers. Giannou et al. (2023) establish looped Transformers as programmable computers. Brösamle & Eckstein (2026) study low-precision softmax Transformers with ordinary and summarized chain-of-thought, allowing logarithmically growing width and depth. Li & Wang (2025) obtain efficient constant-bit simulation with a moving context window and a head–layer/time tradeoff. These results concern different combinations of access, context, dimension, and precision. Our guarantee concerns constant-dimensional recurrent slots and $O(S + p)$ persistent bits, with a specific floating-point arithmetic graph and uniform metadata.