Bounded-Memory Reversible Computation and Housekeeping Dissipation: Garbage Entropy Rate, Active Forgetting, and Finite Reuse

Yining Wu
Independent Researcher
yining.wu@alumni.upenn.edu

FDS-P1 defined the physical accounting interface between formal distinctions and thermodynamic implementation: formal distinctions become task-available physical records only when they are carried, readable, stable, and audited under an accounting boundary. FDS-P2 studies the next step. Reversible computation can avoid immediate Landauer erasure by retaining enough side information to make the enlarged update invertible. Under bounded memory, however, this strategy accumulates garbage records that must be carried, refreshed, synchronized, uncomputed, externalized, compressed, cleaned, or abandoned. This paper strengthens the bounded-memory model by defining garbage entropy rate, accumulated residual irreversibility, nonlinear memory-fill pressure, uncomputation versus irreversible cleanup, externalization as an accounting-boundary shift with latency, active forgetting through lossy compression, and a resource-dependent housekeeping optimization problem. The main theorem states that a bounded-memory system executing sustained reversible updates with positive garbage entropy rate cannot continue indefinitely without uncomputation, cleanup, compression, externalization, task relaxation, memory expansion, or failure. The paper does not refute reversible computation, does not claim that every computational step dissipates $k_{\mathrm{B}}T\ln 2$, and does not derive Landauer's principle. It gives a finite-record accounting model: reversibility changes the time profile and location of dissipation, but bounded memory makes retained inverse information itself a physical burden. Deterministic simulations illustrate correlated garbage entropy rate, memory-fill pressure, cleanup spikes, uncomputation versus cleanup, cleanup scheduling, externalization latency, lossy compression, bounded-memory regimes, and high-rate syndrome-style record turnover as a conservative domain projection.

Scope and Boundary of the Theory. This paper is a physical bridge about bounded memory and record reuse. It does not assert that ideal reversible gates are impossible, that all computation dissipates $k_{\mathrm{B}}T\ln 2$ per operation, that quantum reversible computation is invalid, or that memory management has a universal device-independent power law. It claims only that reversible embeddings avoid immediate erasure by retaining preimage information somewhere inside an accounting boundary, and that sustained reversible updating under finite memory must account for the accumulated side records through uncomputation, refresh, synchronization, externalization, compression, cleanup, task relaxation, memory expansion, or failure.

Claim-status summary. Table I summarizes the central FDS-P2 claims, their epistemic status, and the conditions under which they should be weakened or rejected.

Keywords: reversible computation; bounded memory; garbage entropy rate; garbage records; delayed erasure; Landauer principle; housekeeping dissipation; memory reuse; uncomputation; cleanup; accounting boundary; active forgetting; externalization latency; active finite distinction systems.

INTRODUCTION

From P1 to P2

FDS-P1 established the physical accounting interface for finite distinctions. A formal projection is not heat; a physical many-to-one implementation becomes thermodynamic only when it discards preimage information through reset, overwrite, many-to-one compression, or garbage collection under an accounting boundary [2]. P1's boundary-relative residual irreversibility is

$$L_{\mathcal{A}} = H(X \mid Y, G_{\mathcal{A}}), \tag{1}$$

where X is the pre-update record, Y is the visible post-update record, and $G_{\mathcal{A}}$ is side information retained inside the accounting boundary $\mathcal{A}$.

P2 asks what happens when a system chooses the reversible route. Instead of overwriting preimage information immediately, it embeds the visible computation into a larger invertible map by keeping side records. That avoids immediate erasure relative to $\mathcal{A}$, but it does not eliminate physical accounting. It turns erased bits into garbage records, memory-fill pressure, refresh cost, synchronization cost, externalization cost, or later cleanup.

Thesis

The central thesis is:

TABLE I. Central FDS-P2 claims, epistemic status, and failure or demotion conditions.

Claim	Status	What would weaken or falsify it
Reversible embeddings avoid immediate erasure by retaining preimage information in side records.	Formal / accounting bridge	A non-injective visible map remains invertible without retaining any side record, recovery record, or enlarged state.
Sustained reversible logging accumulates garbage records when the retained garbage process has positive entropy rate.	Conditional theorem	Nontrivial updates with positive garbage entropy rate continue indefinitely while garbage load remains bounded and no uncomputation, compression, or externalization occurs.
Finite memory forces a reuse decision.	Formal finite-capacity claim	A finite memory stores an unbounded sequence of independent or entropy-rate-positive garbage records without overwrite, compression, externalization, or loss.
Uncomputation and cleanup are distinct operations.	Boundary statement	Irreversible cleanup erases garbage without residual irreversibility, or uncomputation succeeds without a retained causal inverse path, protected output copy, or synchronization.
Irreversible garbage cleanup carries boundary-relative Landauer accounting cost.	Physical bridge	Reliable cleanup below generalized Landauer accounting persists after full-boundary audit.
Externalization shifts local memory pressure into write, verification, synchronization, retrieval, maintenance, and latency costs.	Accounting claim	External records impose no storage, verification, synchronization, retrieval, latency, or maintenance burden.
Lossy compression trades memory pressure for residual irreversibility or task distortion.	Rate-distortion bridge	Lossy compression preserves all task-relevant preimage information at zero distortion and zero side cost.
Cleanup scheduling creates a tradeoff between memory pressure, refresh cost, cleanup heat, task loss, and failure risk.	Model class	Cleanup interval, externalization latency, and compression level have no effect on memory pressure, heat, task loss, or failure probability in every controlled implementation.

Reversible computation can postpone erasure, but bounded memory makes postponed history costly.

Reversibility changes when and where dissipation appears. It does not allow a finite system to carry an infinite history for free.

RELATED WORK

FDS core and finite-record physical bridges

The FDS formal core defines active finite distinction systems as systems that maintain boundaries through state-dependent updates under finite representational capacity and resource budgets [1]. It separates the formal layer from the physical bridge: formal preimage loss becomes a thermodynamic heat-rate floor only under Landauer-style implementation assumptions. The core also defines maintenance failure through finite free-energy budgets and the prune–externalize–collapse trichotomy. P1 then defines carriers, accounting boundaries, side records, residual irreversibility, and dissipative projection [2]. P2 is the bounded-memory continuation of that ledger.

Landauer, Bennett, and reversible computation

Landauer's principle links logically irreversible erasure to heat generation in a physical substrate [7]. Bennett showed that computation can be made logically reversible by retaining enough information to reconstruct the input history [8, 9]. Reversible gates and conservative logic further clarify that logical reversibility is possible in principle [10]. Modern information thermodynamics refines these statements for feedback, correlations, stochastic dynamics, and nonideal protocols [11–14, 19, 22]. P2 does not challenge reversible computation. It studies the finite-memory regime in which retained histories become physical burdens.

Garbage collection, external memory, and online systems

Computer systems already implement bounded-memory accounting through garbage collection, checkpointing, logging, compression, paging, and external storage [15]. P2 uses these as engineering analogies, not as proof of a universal power law. The formal claim is narrower: if reversible computation preserves preimages in side records, those records occupy finite carrier capacity until they are uncomputed, compressed, externalized, cleaned, or lost.

REVERSIBLE EMBEDDINGS AND GARBAGE RECORDS

Definition 1 (Visible computation)**.** *Let $f : X \to Y$ be a visible computation or update. If f is many-to-one on a set of positive probability, then $H(X \mid Y) > 0$ for some admissible priors.*

Definition 2 (Reversible embedding)**.** *A reversible embedding of $f : X \to Y$ is an injective map*

$$U : X \times \{0\} \longrightarrow Y \times G \tag{2}$$

such that

$$H(X \mid Y, G) = 0. \tag{3}$$

The variable G is the garbage or side record that carries the preimage information not present in Y alone.

Proposition 1 (Reversible embedding avoids immediate residual irreversibility)**.** *If $U : X \times \{0\} \to Y \times G_{\mathcal{A}}$ is injective inside accounting boundary $\mathcal{A}$, then*

$$L_{\mathcal{A}} = H(X \mid Y, G_{\mathcal{A}}) = 0. \tag{4}$$

No immediate erasure has occurred relative to $\mathcal{A}$.

Proof. Since $(Y, G_{\mathcal{A}})$ selects a unique preimage X, the conditional entropy of X given the enlarged record is zero. By P1's definition of boundary-relative residual irreversibility, $L_{\mathcal{A}} = 0$. □

Remark 1. *The word "garbage" does not mean useless. It means that the record is not part of the visible task output but is required for reversibility. It may be valuable for uncomputation, debugging, recovery, or audit. Its physical cost is that it must be stored and maintained until it is no longer needed.*

GARBAGE ENTROPY RATE AND MEMORY FILL

Definition 3 (Garbage process)**.** *Let $G_{1:t}$ be the side-record process retained to make visible updates reversible over a time window. The ideal independent-increment approximation writes*

$$B_G(t) = \sum_{s=1}^{t} H(X_s \mid Y_s), \tag{5}$$

but the general memory load is the joint entropy

$$B_G(t) = H(G_{1:t} \mid R_t), \tag{6}$$

where R_t denotes task outputs, recovery records, or shared context already retained and not counted as garbage.

Definition 4 (Garbage entropy rate)**.** *The lower asymptotic garbage entropy rate is*

$$r_G = \liminf_{t\to\infty} \frac{1}{t} H(G_{1:t} \mid R_t). \tag{7}$$

The condition $r_G > 0$ means that the garbage history contains a positive rate of new inverse information after correlations and shared recovery context are taken into account.

Definition 5 (Memory fill ratio)**.** *Let M_0 be non-garbage memory usage and $M_{\max}$ the available physical memory capacity. The memory usage and fill ratio are*

$$M_{\text{used}}(t) = M_0 + B_G(t), \qquad \rho_M(t) = \frac{M_{\text{used}}(t)}{M_{\max}}. \tag{8}$$

The fill time is

$$t_{\text{fill}} = \inf\{t : M_{\text{used}}(t) \geq M_{\max}\}. \tag{9}$$

Definition 6 (Memory-fill pressure)**.** *As memory approaches saturation, allocation, indexing, defragmentation, and synchronization can add implementation pressure. A conservative proxy is*

$$C_{\text{alloc}}(\rho_M) = c_{\text{alloc}}[-\log(1 - \rho_M + \epsilon_0)], \tag{10}$$

where $\epsilon_0 > 0$ regularizes the saturation limit. This is an implementation-pressure model, not a universal law.

THE BOUNDED-MEMORY REUSE THEOREM

Theorem 1 (Bounded-memory reuse pressure)**.** *Consider a finite system with memory capacity $M_{\max} < \infty$ executing a sustained sequence of reversible embeddings. Suppose the retained garbage process has entropy rate $r_G > 0$, and suppose the system does not uncompute the garbage, increase memory, externalize records, compress them below the generated entropy rate, relax the task, or discard preimage information. Then the system reaches memory saturation in finite time.*

Proof. By Eq. (7), for any $0 < r < r_G$, there is a sufficiently long interval on which $H(G_{1:t} \mid R_t) \geq rt$ up to sublinear fluctuations. Under the hypotheses, this garbage remains in internal memory, so $M_{\text{used}}(t) = M_0 + H(G_{1:t} \mid R_t)$. Since $M_{\max} < \infty$, there exists finite t such that $M_{\text{used}}(t) \geq M_{\max}$. Saturation is avoided only by using one of the excluded exits: uncomputation, memory expansion, externalization, compression, task relaxation, irreversible discard, or failure. □

Operational coding overhead. The entropy rate r_G gives a lower bound on the lossless storage rate of retained garbage histories. A physical implementation generally requires

$$M_{\text{used}}(t) \geq H(G_{1:t} \mid R_t),$$

with additional overhead for coding inefficiency, indexing, fragmentation, allocation metadata, synchronization, and error correction. The allocation-pressure term $C_{\text{alloc}}(\rho_M)$ models part of this implementation overhead as memory approaches saturation.

Corollary 1 (Linear fill time as special case)**.** *If the garbage increments are approximately independent with mean retained load $\bar{b}_G$, then*

$$t_{\text{fill}} \approx \frac{M_{\max} - M_0}{\bar{b}_G}. \tag{11}$$

For correlated garbage, $\bar{b}_G$ must be replaced by the entropy rate r_G, not by the naive sum of stepwise conditional entropies.

ACCUMULATED RESIDUAL IRREVERSIBILITY

P1's $L_{\mathcal{A}}$ is a per-operation ledger quantity. P2 needs a history-level version.

Definition 7 (Accumulated residual irreversibility)**.** *For a process observed through visible records $Y_{1:t}$ and side records $G_{\mathcal{A},t}$ included inside accounting boundary $\mathcal{A}$, define*

$$\mathcal{I}^{\text{res}}_{\mathcal{A}}(t) = H(X_{1:t} \mid Y_{1:t}, G_{\mathcal{A},t}). \tag{12}$$

This is the history information that remains unrecoverable after all side records currently inside the accounting boundary have been conditioned on.

For a cleanup batch $G_{t_0:t_c}$, the relevant residual irreversibility is

$$L_{\text{clean}}(t_c) = H(G_{t_0:t_c} \mid R_{\mathcal{A},t_c}), \tag{13}$$

where $R_{\mathcal{A},t_c}$ is recovery information still available inside the boundary at cleanup time.

Proposition 2 (Cleanup heat floor)**.** *If a physical cleanup map discards $G_{t_0:t_c}$ with residual irreversibility L_{clean}, then under the Landauer accounting bridge*

$$Q_{\text{clean}} \geq k_{\text{B}} T \ln 2 \; L_{\text{clean}}. \tag{14}$$

For periodic cleanup with period T_{c},

$$\dot{Q}_{\text{clean}} \geq \frac{k_{\text{B}} T \ln 2}{T_{\text{c}}} H(G_{\text{batch}} \mid R_{\mathcal{A}}). \tag{15}$$

Proof. The cleanup map is logically irreversible exactly to the extent that the garbage cannot be reconstructed from remaining records $R_{\mathcal{A}}$. Applying the P1 boundary-relative Landauer bridge to the erased preimage information gives Eq. (14); dividing by the cleanup interval gives Eq. (15). □

Recent work on optimized SRAM erasure further cautions that practical finite-time erasure costs depend strongly on the physical memory model and driving protocol, rather than following a universal linear speed–dissipation rule [20].

UNCOMPUTATION VERSUS CLEANUP

A common objection is that Bennett uncomputation can remove garbage without paying an erasure cost. P2 accepts this. The distinction is that uncomputation is not cleanup.

Definition 8 (Uncomputation)**.** *Uncomputation is a reversible operation that maps*

$$(Y, G) \longrightarrow (Y, 0) \tag{16}$$

by reversing a still-available computation path while preserving an allowed copy of the task output Y.

Definition 9 (Irreversible cleanup)**.** *Cleanup is a physical operation that maps*

$$G \longrightarrow 0 \tag{17}$$

without retaining enough recovery information to reconstruct G. Its residual irreversibility is $H(G \mid R_{\mathcal{A}})$.

Proposition 3 (Conditions for uncomputation)**.** *Uncomputation succeeds only if the garbage remains causally recoverable, the inverse path or equivalent recovery information remains available, the output record that must survive is protected, and synchronization has not been lost.*

Reason. Reversing a computation requires an inverse map and the records needed to select the correct inverse branch. If the path is not recoverable, if the output would be destroyed, or if the system has lost synchronization with the state to be reversed, the operation is no longer uncomputation. It becomes cleanup, lossy compression, or failure. □

Remark 2. *Uncomputation removes garbage by reversing a still-available computation path; cleanup removes garbage by discarding information. Only cleanup carries residual irreversibility. But uncomputation still requires physical carriers, clocking, output protection, and synchronization.*

COMPRESSION, ACTIVE FORGETTING, AND TASK LOSS

Finite systems can choose to forget. This is not always failure. It can be an active housekeeping policy.

Definition 10 (Lossy garbage compression)**.** *A lossy compression policy maps garbage G to a compressed record $\tilde{G}$. Its residual irreversibility and task distortion are*

$$L_{\mathrm{comp}} = H(G \mid \tilde{G}, R_{\mathcal{A}}), \tag{18}$$

$$D_{\mathrm{task}} = \mathbb{E}[d_{\mathrm{task}}(G, \tilde{G})]. \tag{19}$$

Compression can produce heat and residual irreversibility, but reduce memory-fill pressure and refresh cost. Active forgetting can therefore be optimal when

$$\Delta Q_{\mathrm{refresh}} + \Delta C_{\mathrm{alloc}} + \mu\, \Delta P_{\mathrm{fail}} > Q_{\mathrm{comp}} + \lambda D_{\mathrm{task}}, \tag{20}$$

This is a policy inequality, not a universal biological law. It says that retaining every distinction is not always thermodynamically or operationally optimal.

Proposition 4 (Forgetting can be a finite-memory policy)**.** *If the physical cost of maintaining detailed garbage records exceeds their task value plus the cost of lossy compression, then a finite system can reduce total expected housekeeping cost by intentionally discarding detail.*

Lossy compression creates residual irreversibility at the information-accounting level, but its thermodynamic cost depends on physical implementation. An abstract lossy map is not heat by itself. It becomes a dissipative operation only when implemented through physical overwrite, reset, many-to-one compression, or garbage collection inside an accounting boundary.

EXTERNALIZATION AS ACCOUNTING-BOUNDARY SHIFT

Externalization moves retained records from local memory into an external carrier. In P1 language, it changes the accounting boundary:

$$\mathcal{A}_{\mathrm{local}} \longrightarrow \mathcal{A}_{\mathrm{coupled}} = \mathcal{A}_{\mathrm{local}} \cup \mathcal{A}_{\mathrm{ext}}. \tag{21}$$

Local memory pressure falls, but new costs appear:

$$C_{\mathrm{ext}} = C_{\mathrm{write}} + C_{\mathrm{verify}} + C_{\mathrm{sync}} + C_{\mathrm{retrieve}} + C_{\mathrm{maint}} + C_{\mathrm{latency}}. \tag{22}$$

The O2 interface is timing. An external record exists, but it is task-useful only if retrieval and synchronization fit inside the task window:

$$T_{\mathrm{retrieve}} + T_{\mathrm{sync}} + T_{\mathrm{verify}} \le \tau_{\mathrm{task}}. \tag{23}$$

If Eq. (23) fails, the external record remains physically real but is not operationally available for that task. Externalization shifts the accounting boundary; it does not remove timing constraints.

HOUSEKEEPING OPTIMIZATION

A bounded-memory system chooses a housekeeping policy π: when to uncompute, when to clean, how much to compress, when to externalize, and when to relax task fidelity. A FDS-aligned long-run objective is

$$J(\pi) = \langle \dot{Q}_{\mathrm{hk}} \rangle_\pi + \lambda_\ell(\Phi) \langle \ell_{\mathrm{task}} \rangle_\pi + \nu(\Phi) \langle \Delta_{\mathrm{residual}} \rangle_\pi + \mu(\Phi) P_{\mathrm{fail}}. \tag{24}$$

subject to

$$M_{\mathrm{used}}(t) \le M_{\max} \tag{25}$$

and the FDS resource update constraint

$$\Phi_{t+1} \le \Phi_t + \int_t^{t+\tau} \dot{F}_{\mathrm{in}}(s) ds - \int_t^{t+\tau} \dot{Q}_{\mathrm{hk}}(s) ds - D_\ell(\ell_t). \tag{26}$$

Here ℓ_{task} is task loss, $\Delta_{\mathrm{residual}}$ is residual irreversibility or task-relevant forgotten information, and P_{fail} is failure probability. The weights are shadow prices that may depend on the remaining resource budget Φ. A simple failure-avoidance weight is

$$\mu(\Phi) = \mu_0 + \frac{\mu_1}{\Phi - \Phi_{\mathrm{crit}} + \epsilon_0}, \tag{27}$$

for $\Phi > \Phi_{\mathrm{crit}}$. Resource depletion changes the relative shadow prices of energy, task fidelity, residual irreversibility, and failure risk.

NUMERICAL MODELS AND SIMULATIONS

The simulations are deterministic synthetic demonstrations. They are not fits to physical memory devices, detector data, quantum experiments, AI benchmarks, or biological records. They provide reproducible model diagrams for the P2 accounting layer. All figures and CSV outputs are generated by `code/generate_results.py`.

Simulation summary

Figure 1 shows why the theorem uses entropy rate rather than a naive stepwise sum. Figure 2 illustrates

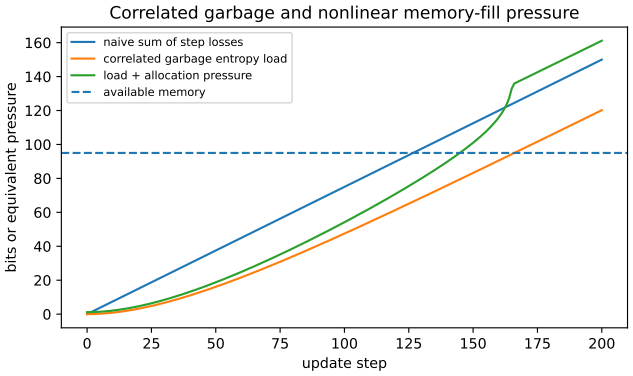

FIG. 1. Correlated garbage and nonlinear memory-fill pressure. The naive sum of step losses can overstate memory load when garbage is correlated. The entropy-rate load is lower, but allocation pressure rises nonlinearly as memory approaches saturation.

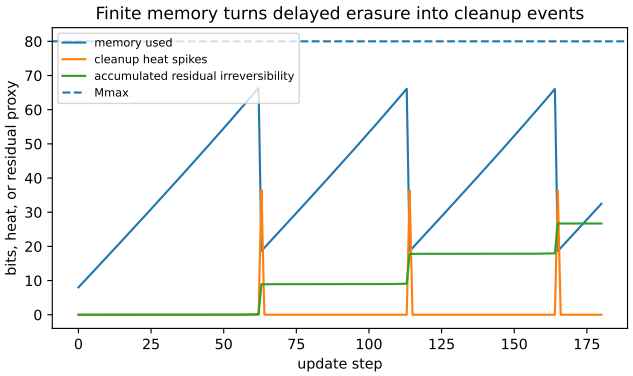

FIG. 2. Finite memory turns delayed erasure into cleanup events. Reversible logging fills memory; threshold cleanup produces heat spikes and can leave accumulated residual irreversibility.

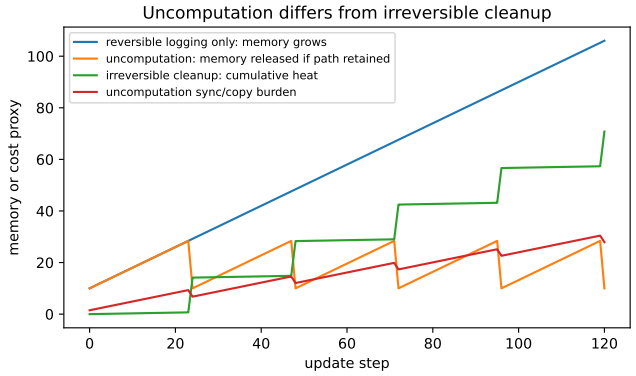

FIG. 3. Uncomputation differs from irreversible cleanup. Logging alone grows memory. Uncomputation can release memory if the inverse path and output copy are retained, but it carries synchronization and output-protection burden. Cleanup discards garbage and accumulates heat.

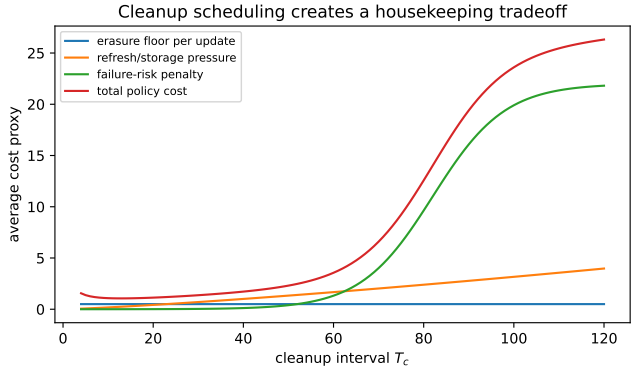

FIG. 4. Cleanup scheduling creates a housekeeping tradeoff. Frequent cleanup carries repeated cleanup overhead, while delayed cleanup increases refresh and storage pressure and failure-risk penalty. The optimum depends on device and task weights, not on a universal cleanup interval.

dynamic residual irreversibility. Figure 3 separates uncomputation from cleanup. Figure 4 illustrates cleanup scheduling. Figure 5 adds externalization latency. Figure 6 shows lossy compression as active forgetting. Figure 7 gives a bounded-memory regime diagram. Figure 8 gives a conservative syndrome-style record-turnover projection.

DOMAIN PROJECTION: SYNDROME-STYLE RECORDS

High-rate error-correction systems repeatedly generate finite classical records: syndrome bits, detector clicks, parity checks, log records, or controller states. Recent below-threshold surface-code experiments show that syndrome production, decoding, reset, and real-time latency are already central engineering constraints in superconducting QEC systems [21]. P2 does not claim a no-go theorem for quantum computation. It says only that if a record-bearing architecture repeatedly generates, stores, decodes, resets, or externalizes high-rate side information, the accounting categories of P2 apply: garbage entropy rate, memory fill, uncomputation where possible, irreversible cleanup where necessary, externalization latency, and housekeeping dissipation.

For a syndrome-style record stream with N_s records per cycle, b_s bits per record, and cycle time τ_c, the irreversible reuse lower bound is

$$P_{\mathrm{erase}} \geq \frac{k_{\mathrm{B}} T \ln 2}{\tau_c} N_s b_{\mathrm{erase}}. \tag{28}$$

Actual power may be dominated by readout, control, classical processing, routing, or cooling overhead rather than the ideal Landauer floor. P2's purpose is to keep

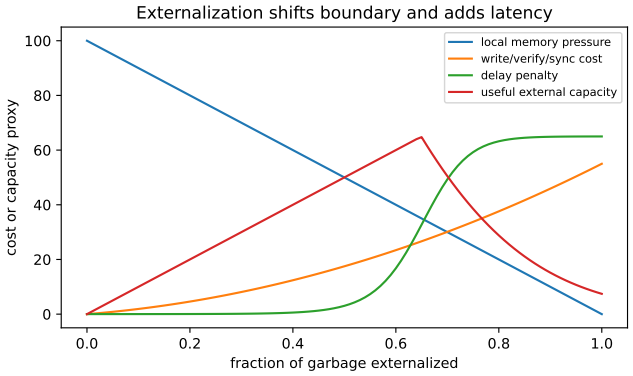

FIG. 5. Externalization shifts boundary and adds latency. Moving garbage to external memory reduces local pressure but increases write, verification, synchronization, retrieval, and delay penalties. Beyond the task window, external records become less operationally useful.

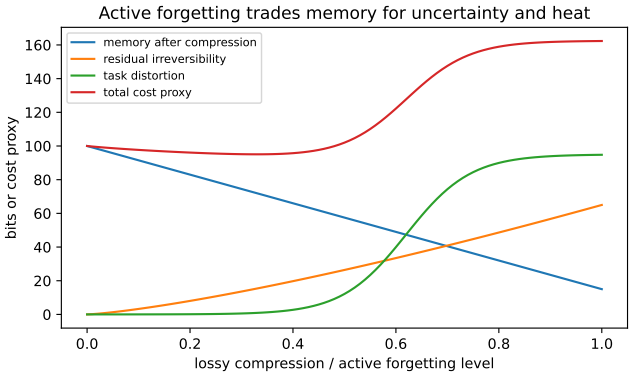

FIG. 6. Active forgetting trades memory for uncertainty and heat. Lossy compression reduces memory and refresh burden but creates residual irreversibility and task distortion. In some regimes forgetting is cheaper than maintaining all detail.

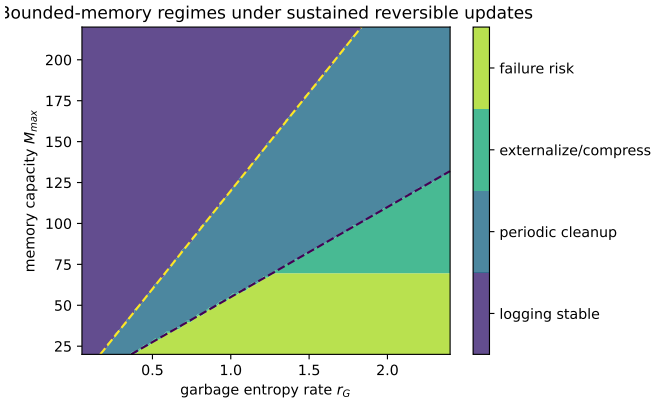

FIG. 7. Bounded-memory regimes under sustained reversible updates. The diagram is a synthetic phase portrait in garbage entropy rate and memory capacity. Regions indicate stable logging, periodic cleanup, externalization/compression dominance, and failure risk.

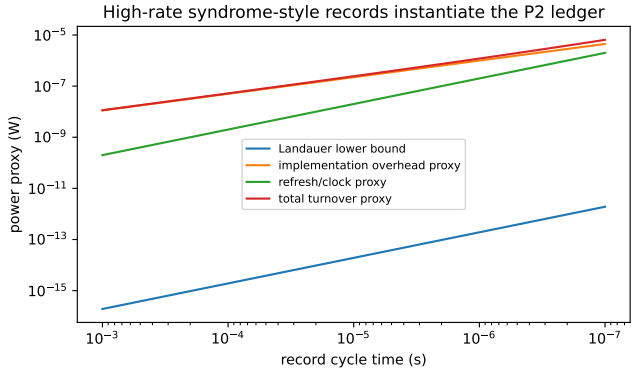

FIG. 8. High-rate syndrome-style records instantiate the P2 ledger. The plot is not a quantum-computing no-go theorem. It illustrates that repeated finite-record turnover has a lower-bound erasure term plus implementation-dependent refresh, clocking, and processing overhead.

these terms in the same finite-record ledger, not to reduce them to one constant. This section is a domain projection of the P2 ledger, not a no-go theorem for quantum computation and not a claim that all QEC architectures are thermodynamically impossible. It only identifies syndrome production, reset, decoding, latency, and finite-memory turnover as instances of the bounded-record accounting problem.

EXPERIMENTAL AND ENGINEERING PROTOCOLS

Protocol 1 (Entropy-rate garbage audit)**.** *Instrument a reversible or logging-based computation and estimate $H(G_{1:t} \mid R_t)/t$ rather than summing per-step losses. P2 predicts that correlated garbage reduces load relative to the naive sum but positive entropy rate still fills bounded memory.*

Protocol 2 (Uncomputation versus cleanup)**.** *Compare a reversible uncomputation schedule with an irreversible cleanup schedule under the same task. Measure memory load, synchronization overhead, output-copy burden, and heat spikes. P2 predicts different ledgers for the two operations.*

Protocol 3 (Externalization latency test)**.** *Move increasing fractions of side records into an external store. Measure local memory relief, write/verify/sync/retrieval cost, and delay relative to τ_{task}. P2 predicts that externalization fails as a task resource when retrieval latency exceeds the task window.*

Protocol 4 (Active forgetting tradeoff)**.** *Vary compres-*

sion level for retained garbage records. Measure memory load, refresh cost, residual irreversibility, task distortion, and failure risk. P2 predicts regimes where lossy forgetting reduces total housekeeping cost, and regimes where it destroys task fidelity.

LIMITATIONS AND FALSIFICATION

P2 is not a universal theory of computation. It is a finite-record accounting model for sustained reversible updating under bounded memory. It does not deny ideal reversible gates, reversible quantum evolution, or reversible algorithms under their hypotheses. It does not claim that every step dissipates $k_{\mathrm{B}}T\ln 2$. It also does not claim that all garbage must be irreversibly cleaned; uncomputation is a valid strategy when its conditions hold.

The strong version of FDS-P2 would be weakened or rejected by any of the following:

1. bounded-memory reversible systems sustain unbounded positive-entropy garbage histories without uncomputation, compression, externalization, cleanup, memory expansion, task relaxation, or failure;
2. irreversible garbage cleanup erases positive residual information below generalized Landauer accounting after full-boundary audit;
3. retained side records impose no carrier, refresh, synchronization, retrieval, indexing, or latency burden;
4. uncomputation succeeds after the inverse path, output copy, or synchronization records have been destroyed;
5. externalization always remains task-useful regardless of retrieval and synchronization latency;
6. lossy compression preserves all task-relevant preimage information at zero distortion in every nontrivial task;
7. cleanup interval, memory fill ratio, and garbage entropy rate have no measurable effect on heat, latency, failure risk, or task loss in every controlled implementation.

CONCLUSION

P1 gave the accounting ledger. P2 shows why finite memory makes ledger reuse costly. Reversible computation postpones erasure by preserving the past. Bounded memory makes that preservation itself a physical burden. The system must either uncompute while the inverse path remains available, preserve and refresh side records, externalize them into a coupled boundary, compress and accept residual irreversibility, relax the task, or clean and pay the erasure account.

The compact statement is:

> *Reversibility preserves inverse information; bounded memory makes preserved inverse information compete with refresh, allocation, synchronization, latency, task fidelity, and failure risk. Forgetting becomes a policy, not merely a failure.*

This is the physical bridge needed before a full FDS treatment of entropy production in active finite systems and finite-memory Second-Law channels.

Notation Summary

Simulation Parameters

The simulations are deterministic and use fixed synthetic parameters in `code/generate_results.py`. Figure 1 uses correlated garbage load and an allocation-pressure proxy. Figure 2 uses threshold cleanup with residual irreversibility. Figure 3 compares logging only, uncomputation, and cleanup. Figure 4 varies cleanup interval. Figure 5 varies externalized fraction and latency. Figure 6 varies lossy compression level. Figure 7 uses a synthetic regime classifier in $(r_G, M_{\max})$. Figure 8 uses high-rate record-turnover proxies. No proprietary, device, biological, quantum, or human-subject data are used.

Reproducibility Checklist

1. Code availability: all simulation code is included in the replication package.
2. Deterministic execution: the random seed is fixed where randomness is used.
3. Figure reproduction: run `python code/generate_results.py`; the script regenerates all figures and CSV outputs.
4. Data status: all numerical outputs are synthetic demonstrations generated from the stated model.
5. Platform independence: the code uses standard Python scientific libraries.

Code Availability

The simulation code used to generate Figs. 1–8 is included in the accompanying replication package under

TABLE II. FDS-P2 notation summary.

Symbol	Meaning
X	pre-update physical or computational record
Y	visible output or post-update record
G	garbage / side record carrying inverse information
$G_{\mathcal{A}}$	side record retained inside accounting boundary $\mathcal{A}$
R_t	retained task output, recovery context, or already-counted record
$L_{\mathcal{A}}$	boundary-relative residual irreversibility $H(X \mid Y, G_{\mathcal{A}})$
$B_G(t)$	garbage load, generally $H(G_{1:t} \mid R_t)$
r_G	lower garbage entropy rate $\liminf t^{-1} H(G_{1:t} \mid R_t)$
M_{used}	memory used by active records and garbage
$M_{\max}$	finite memory capacity
ρ_M	memory fill ratio $M_{\text{used}}/M_{\max}$
C_{alloc}	allocation / fragmentation / indexing pressure as $\rho_M \to 1$
L_{clean}	residual irreversibility of a cleanup batch
$\dot{Q}_{\text{hk}}$	housekeeping power: refresh, cleanup, sync, externalization, compression, etc.
ℓ_{task}	task loss
Φ	FDS resource or free-energy budget
τ_{task}	task window for useful retrieval or synchronization

`code/generate_results.py`. Running the script regenerates all figures (PDF and PNG) and CSV tables in a single pass.

AI Assistance Disclosure

AI-assisted tools were used for language polishing, structural feedback, LaTeX drafting support, and code-debugging assistance. The author reviewed and edited all content and remains responsible for all claims, references, simulations, and conclusions. No AI system is listed as an author.

[1] Y. Wu, "Active Finite Distinction Systems: A Formal Core for Boundary Maintenance under Finite Capacity," Zenodo (2026), doi:10.5281/zenodo.20158923.

[2] Y. Wu, "Physical Distinction Carriers and Erasure Maps: Accounting Boundaries, Distinction-to-Noise Ratio, and Thermodynamic Implementation," Zenodo (2026), doi:10.5281/zenodo.20251854.

[3] Y. Wu, "Time as Irreversible Distinction Update: Finite Records, Causal Ordering, and Register-Time Collapse," Zenodo (2026), doi:10.5281/zenodo.20249369.

[4] Y. Wu, "Capacity Overflow, Effective Stochasticity, and Phase-B Invariants: Critical Deficit, Markov Closure, and Invariant Selection under Finite Projection," Zenodo (2026), doi:10.5281/zenodo.20250367.

[5] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal **27**, 379–423 and 623–656 (1948).

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley (2006).

[7] R. Landauer, "Irreversibility and heat generation in the computing process," IBM Journal of Research and Development **5**, 183–191 (1961).

[8] C. H. Bennett, "Logical reversibility of computation," IBM Journal of Research and Development **17**, 525–532 (1973).

[9] C. H. Bennett, "The thermodynamics of computation–a review," International Journal of Theoretical Physics **21**, 905–940 (1982).

[10] E. Fredkin and T. Toffoli, "Conservative logic," International Journal of Theoretical Physics **21**, 219–253 (1982).

[11] T. Sagawa, "Thermodynamics of information processing in small systems," Progress of Theoretical Physics **127**, 1–56 (2012).

[12] J. M. R. Parrondo, J. M. Horowitz, and T. Sagawa, "Thermodynamics of information," Reviews of Modern Physics **87**, 45–67 (2015).

[13] D. H. Wolpert, "The stochastic thermodynamics of computation," Journal of Physics A: Mathematical and Theoretical **52**, 193001 (2019).

[14] P. Chattopadhyay, A. Misra, T. Pandit, and G. Paul, "Landauer principle and thermodynamics of computation," Reports on Progress in Physics **88**, 086001 (2025).

[15] P. R. Wilson, "Uniprocessor garbage collection techniques," in *International Workshop on Memory Management*, Lecture Notes in Computer Science **637**, 1–42 (1992).

[16] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," Physical Review A **52**, R2493–R2496 (1995).

[17] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. thesis, California Institute of Technology (1997), arXiv:quant-ph/9705052.

[18] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," Physical Review A **86**, 032324 (2012).

[19] D. H. Wolpert, J. Korbel, C. W. Lynn, F. Tasnim, J. A. Grochow, G. Kardes, J. B. Aimone, V. Balasubramanian, E. De Giuli, D. Doty, N. Freitas, M. Marsili, T. E. Ouldridge, A. W. Richa, P. M. Riechers, E. Roldán, B. Rubenstein, Z. Toroczkai, and J. Paradiso, "Is stochastic thermodynamics the key to understanding

the energy costs of computation?" Proceedings of the National Academy of Sciences **121**, e2321112121 (2024), doi:10.1073/pnas.2321112121.

[20] T. Basile and K. Proesmans, "Learning optimal erasure of a Static Random Access Memory," arXiv:2411.02044 (2024), doi:10.48550/arXiv.2411.02044.

[21] Google Quantum AI and Collaborators, "Quantum error correction below the surface code threshold," Nature **638**, 920–926 (2025), doi:10.1038/s41586-024-08449-y.

[22] S. Aimet, M. Tajik, G. Tournaire, P. Schüttelkopf, J. Sabino, S. Sotiriadis, G. Guarnieri, J. Schmiedmayer, and J. Eisert, "Experimentally probing Landauer's principle in the quantum many-body regime," Nature Physics **21**, 1326–1331 (2025), doi:10.1038/s41567-025-02930-9.