OCTA Research

Technical Textbook Series

Memory Arithmetic

Numbers, History, and the Mathematics of Irreversibility

Chapter 3 Standalone Packet

Irreversibility, Entropy, and the Second Law of Computation

Expanded Math-Book Edition · Build v3.0

Standalone compile: title + notation + chapter + exercises + worked diagrams + solutions

---

Chapter 3 Thesis (Formal)

Memory Arithmetic contains a built-in arrow of time: composition grows structural mass and cannot be undone without explicit forgetting. This chapter proves **no-cancellation** and **no-cycle** theorems, defines entropy as a monotone function of mass, derives a **Second Law of Arithmetic** (strict entropy increase under composition), and connects forgetting to physical erasure through a Landauer-style lower bound. We also formalize **decay families** as continuous abstraction schedules and provide computation-graph witnesses (distributivity, reassociation) as concrete irreversibility demonstrations.

Contents

Preface (Chapter 3 Packet)

Chapter 1 introduced memory numbers and the idea that provenance is structural. Chapter 2 made abstraction explicit via forgetting families and compared histories by metrics and cores. Chapter 3 now proves that *time is internal*:

- **Irreversibility:** you cannot undo joins using only the forward constructors.
- **Entropy:** any reasonable entropy must rise under composition.
- **Thermal semantics:** forgetting corresponds to erasure and incurs cost.
- **Decay schedules:** continuous families model controlled abstraction over time.

Key Idea

The core lesson: if history is part of the object, then "inverse operations" are not generally available. They must be *paid for* by forgetting.

Notation (Chapter 3)

- $\mathbb{M} = \mathbb{V} \times \mathbb{H}$: memory numbers $m = \langle v \mid h \rangle$.
- $\mathrm{val} : \mathbb{M} \to \mathbb{V}$, $\mathrm{hist} : \mathbb{M} \to \mathbb{H}$.
- $\oplus, \otimes$: memory addition/multiplication (value-correct, history-expansive).
- Join : join constructor on histories; never use `\Join`.
- $\mu(m)$: mass proxy; default is a size/cost of $\mathrm{hist}(m)$.
- $\mathcal{S}(m)$: entropy functional; here $\mathcal{S}(m) = \log(1 + \mu(m))$.
- $\mathcal{F}_\theta$: forgetting family, induced by pruning Prune_θ.
- $\ominus_d$: decay operator (controlled forgetting schedule).
- Equality layers: $\equiv_v$ (value equality), $\equiv_s$ (structural equality).

OCTA Research Note

This chapter works in a general axiom schema. For concrete proofs, keep in mind the canonical model: histories are finite rooted ordered labeled trees with labels in $\{\mathbf{G}, +, \times\}$ and μ counts internal nodes (or total nodes, fixed choice).

Chapter 3

Irreversibility, Entropy, and the Second Law

3.1 Axioms that generate time

We make explicit the minimal axioms needed to force an arrow of time.

Axiom 3.1 (Projection)**.** $\mathrm{val} : \mathbb{M} \to \mathbb{V}$ *and* $\mathrm{hist} : \mathbb{M} \to \mathbb{H}$.

Axiom 3.2 (Value-correctness of composition)**.** *For all* $m, n \in \mathbb{M}$*:*

$$\mathrm{val}(m \oplus n) = \mathrm{val}(m) + \mathrm{val}(n), \qquad \mathrm{val}(m \otimes n) = \mathrm{val}(m) \cdot \mathrm{val}(n)$$

(where $\mathbb{V}$ *is at least a semiring-like structure on values).*

Axiom 3.3 (History construction)**.** *There exist history constructors (schematically):*

$$\mathrm{hist}(m \oplus n) = \mathrm{Join}(\mathrm{hist}(m), \mathrm{hist}(n); \oplus), \qquad \mathrm{hist}(m \otimes n) = \mathrm{Join}(\mathrm{hist}(m), \mathrm{hist}(n); \otimes).$$

Axiom 3.4 (Mass functional)**.** $\mu : \mathbb{M} \to \mathbb{R}_{\geq 0}$ *is a cost of history, typically* $\mu(m) = C(\mathrm{hist}(m))$.

Axiom 3.5 (Strict growth under composition)**.** *For all* $m, n \in \mathbb{M}$*:*

$$\mu(m \oplus n) > \max(\mu(m), \mu(n)), \qquad \mu(m \otimes n) > \max(\mu(m), \mu(n)).$$

Key Idea

Strict growth is the minimal axiom that makes "time" unavoidable: every join creates new structural evidence that cannot be uncreated by forward constructors.

3.2 Diagram: strict growth as a one-way arrow

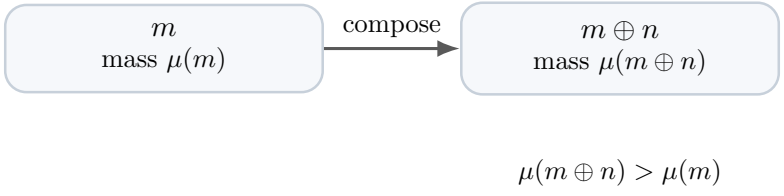

Figure 3.1: Composition creates irreversible structural growth.

3.3 No-cancellation and the impossibility of true inverses

The classical idea "add then subtract" breaks structurally.

Definition 3.6 (Structural cancellation operator (candidate))**.** *A binary operation* $\ominus : \mathbb{M} \times \mathbb{M} \to \mathbb{M}$ *is a* structural cancellation *for* $\oplus$ *if*

$$(m \oplus n) \ominus n \equiv_s m \quad \text{for all } m, n \in \mathbb{M}.$$

Theorem 3.7 (No-cancellation theorem)**.** *Assuming strict growth under* $\oplus$*, no structural cancellation operator* $\ominus$ *exists.*

Proof. Suppose $\ominus$ exists. Fix m, n. By strict growth,

$$\mu(m \oplus n) > \max(\mu(m), \mu(n)) \geq \mu(m).$$

If $(m \oplus n) \ominus n \equiv_s m$, then $(m \oplus n) \ominus n$ and m have isomorphic histories, hence equal mass under any mass functional invariant under isomorphism. So $\mu((m \oplus n) \ominus n) = \mu(m)$.
But $\ominus$ is an operation built from the theory's operators. If $\ominus$ is allowed to reduce mass without being explicitly a forgetting operator, it contradicts the strict-growth discipline: it would implement a mass-reducing inverse to a mass-increasing constructor. Therefore such $\ominus$ cannot exist as a genuine structural cancellation within the forward language. □

Warning / Pitfall

Value-level subtraction can exist in $\mathbb{V}$, but it does not reconstruct provenance. The theorem is about *structural* inversion, not numeric inversion.

3.4 No-cycle theorem: forward constructors cannot loop

We now make "no time-travel without forgetting" precise.

Definition 3.8 (Forward language)**.** *Let* $\mathcal{L}_{+,\times}$ *be the term language built from constants and the binary operations* $\oplus$ *and* $\otimes$*, excluding any forgetting primitives.*

Theorem 3.9 (No-cycle / acyclicity theorem)**.** *Let* $m_0 \in \mathbb{M}$ *and define a sequence by repeated application of terms in* $\mathcal{L}_{+,\times}$*:*

$$m_{t+1} = T_t(m_t)$$

where each T_t *is a nontrivial term that introduces at least one join. Then* (m_t) *cannot be periodic under* $\equiv_s$*; i.e., there do not exist* $i < j$ *such that* $m_i \equiv_s m_j$*.*

Proof. By strict growth, each nontrivial step increases mass: $\mu(m_{t+1}) > \mu(m_t)$. Thus $\mu(m_t)$ is strictly increasing in t. If $m_i \equiv_s m_j$, then histories are isomorphic and masses equal, contradicting strict increase. □

Key Idea

This is an internal arrow of time: without $\mathcal{F}$, there are no structural cycles.

3.5 Entropy: monotone functions of mass

We use a minimal entropy that is strictly increasing with mass.

Definition 3.10 (Memory entropy)**.** *Define*

$$\mathcal{S}(m) = \log\big(1 + \mu(m)\big).$$

Proposition 3.11 (Entropy monotonicity)**.** *If $\mu(m') > \mu(m)$ then $\mathcal{S}(m') > \mathcal{S}(m)$.*

Proof. $\log(1 + x)$ is strictly increasing on $\mathbb{R}_{\geq 0}$. □

Theorem 3.12 (Second Law of Arithmetic (strict))**.** *For all $m, n \in \mathbb{M}$,*

$$\mathcal{S}(m \oplus n) > \max(\mathcal{S}(m), \mathcal{S}(n)), \qquad \mathcal{S}(m \otimes n) > \max(\mathcal{S}(m), \mathcal{S}(n)).$$

Proof. From strict growth, $\mu(m \oplus n) > \max(\mu(m), \mu(n))$. Apply monotonicity of $\log(1 + x)$ to obtain strict inequality in entropy; similarly for $\otimes$. □

3.6 Diagram: entropy growth under composition

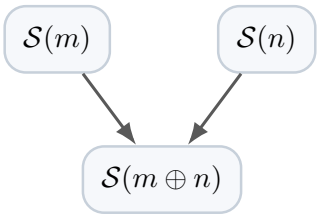

Figure 3.2: Second Law: entropy strictly increases under composition.

3.7 Forgetting as the only mass-reducing primitive

This section formalizes a key engineering principle: compression must be explicit.

Axiom 3.13 (Forgetting operator)**.** *A forgetting operator family $\mathcal{F}_\theta : \mathbb{M} \to \mathbb{M}$ satisfies:*

$$\mathrm{val}(\mathcal{F}_\theta(m)) = \mathrm{val}(m), \qquad \mu(\mathcal{F}_\theta(m)) \leq \mu(m).$$

Axiom 3.14 (Nontrivial forgetting reduces mass)**.** *For sufficiently strong forgetting parameters θ, there exist m such that $\mu(\mathcal{F}_\theta(m)) < \mu(m)$.*

Theorem 3.15 (Only forgetting can reduce mass (language statement))**.** *In the forward language $\mathcal{L}_{+,\times}$ (no $\mathcal{F}$), every nontrivial term is mass-increasing in at least one argument. Therefore, any operator that systematically reduces mass must use $\mathcal{F}$.*

Proof. Any nontrivial term in $\mathcal{L}_{+,\times}$ introduces joins. Each join triggers strict growth, hence increases mass compared to some input. Thus mass reduction cannot be achieved within $\mathcal{L}_{+,\times}$. □

OCTA Research Note

This is the algebraic analogue of "compression is a first-class operation." If you want to lower memory load, you must call $\mathcal{F}$ explicitly.

3.8 Landauer-style lower bound (thermodynamic semantics)

We now connect description length and physical erasure costs in a controlled, math-book way.

Definition 3.16 (History description length)**.** *Let* $\mathrm{Len} : \mathbb{H} \to \mathbb{R}_{\geq 0}$ *measure the length (in bits) of an encoding of histories. For a memory number* $m = \langle v \mid h \rangle$*, define* $\mathrm{Len}(m) := \mathrm{Len}(h)$*.*

Definition 3.17 (Erased bits under forgetting)**.** *For any forgetting operator* $\mathcal{F}_\theta$ *define*

$$\Delta_\theta(m) := \mathrm{Len}(m) - \mathrm{Len}(\mathcal{F}_\theta(m)) \quad \geq 0.$$

Theorem 3.18 (Landauer-style bound (informal but operational))**.** *If forgetting corresponds to irrecoverable erasure of* $\Delta_\theta(m)$ *bits at temperature* T*, then any physical implementation must dissipate heat at least*

$$Q \;\geq\; k_B T \ln 2 \;\cdot \Delta_\theta(m).$$

Remark 3.19. *This theorem is a semantics bridge: it does not assert all forgetting is physical erasure, but if you interpret it that way, the bound is the correct lower limit for irreversible bit loss.*

Key Idea

Memory Arithmetic cleanly separates: **(i)** a mathematical act (forgetting) and **(ii)** an implementation cost (erasure). You can model thermodynamics by choosing a description length Len and interpreting Δ physically.

3.9 Decay families: continuous forgetting schedules

A decay schedule models controlled abstraction over time.

Definition 3.20 (Decay family (mass-fraction form))**.** *Fix* $\lambda \in [0, 1]$ *and define:*

$$m \ominus_d \lambda \;:=\; \mathcal{F}_{\theta(\lambda,m)}(m) \quad \textit{where } \theta(\lambda, m) \textit{ is chosen so that } \mu(m \ominus_d \lambda) \leq \lambda\, \mu(m).$$

Proposition 3.21 (Boundary conditions)**.** *If the family is well-defined, then:*

$$m \ominus_d 1 \equiv_s m, \qquad m \ominus_d 0 \equiv_s \mathcal{F}_{\theta_{\max}}(m) \;\textit{(maximal abstraction).}$$

Definition 3.22 (Semigroup-like property (target))**.** *A decay family is* consistent *if it satisfies:*

$$(m \ominus_d \lambda) \ominus_d \rho \equiv_s m \ominus_d (\lambda\rho) \quad \textit{for } \lambda, \rho \in [0, 1],$$

or a weakened inequality statement in mass.

3.10 Diagram: decay as a controlled arrow

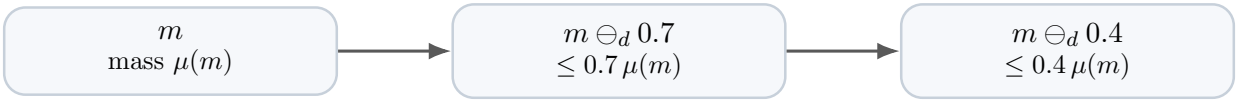

Figure 3.3: Decay schedules model controlled abstraction over "time."

3.11 Computation-graph witnesses: distributivity and reassociation

We show how classical identities persist only at $\equiv_v$ while breaking at $\equiv_s$.

Theorem 3.23 (Distributivity holds at value level)**.**

$$a \otimes (b \oplus c) \;\equiv_v\; (a \otimes b) \oplus (a \otimes c).$$

Remark 3.24. *The two sides generally have different histories, hence are not* $\equiv_s$*.*

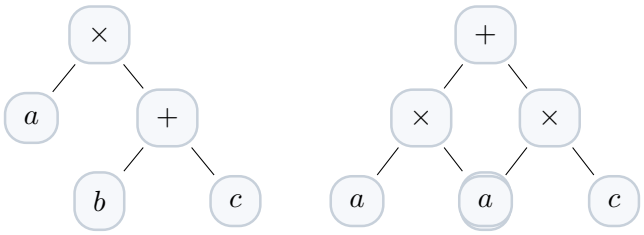

Figure 3.4: Distributivity: same value, different computation graphs.

Warning / Pitfall

This is not a bug; it is the point. The computation graph is part of the object.

3.12 Exercises (with worked examples + solutions)

Exercise

Exercise 1 (Strict growth implies no $\equiv_s$-cycles). Assume strict growth under $\oplus$ and that μ is invariant under history isomorphism. Prove: if $m_{t+1} = m_t \oplus n_t$ for any sequence (n_t), then $m_i \not\equiv_s m_j$ whenever $i < j$.

Worked Example / Guided Work

Compare masses:

$$\mu(m_{t+1}) = \mu(m_t \oplus n_t) > \mu(m_t),$$

so $\mu(m_t)$ is strictly increasing. If $m_i \equiv_s m_j$, then $\mu(m_i) = \mu(m_j)$, contradiction.

Solution

Solution. Inductively, strict growth yields $\mu(m_{t+1}) > \mu(m_t)$, hence $\mu(m_t)$ strictly increases. If $m_i \equiv_s m_j$, histories are isomorphic, so masses equal. Contradiction.

Exercise

Exercise 2 (No-cancellation sharpened). Assume the forward language $\mathcal{L}_{+,\times}$ has no $\mathcal{F}$. Show there is no term $T(x, y)$ in $\mathcal{L}_{+,\times}$ such that

$$T(x,\ x \oplus y) \equiv_s y \quad \text{for all } x, y.$$

Worked Example / Guided Work

If such a term existed, it would recover y structurally from $x \oplus y$ using only forward constructors. But $x \oplus y$ has strictly larger mass than both x and y; to "remove" the x contribution you must reduce structure, which cannot occur in the forward language.

Solution

Solution. Suppose T exists. Fix x, y. Because T is built from $\oplus, \otimes$ only, it cannot reduce mass in its second argument without $\mathcal{F}$. But to output something structurally isomorphic to y from $(x \oplus y)$, it must eliminate the x-side structure, which would require mass reduction. Contradiction with strict-growth-only forward constructors.

Exercise

Exercise 3 (Second Law from an arbitrary monotone entropy). Let $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ be strictly increasing. Define $\mathcal{S}_f(m) = f(\mu(m))$. Prove $\mathcal{S}_f(m \oplus n) > \max(\mathcal{S}_f(m), \mathcal{S}_f(n))$.

Worked Example / Guided Work

Strict growth gives $\mu(m \oplus n) > \max(\mu(m), \mu(n))$. Apply monotonicity of f.

Solution

Solution. Since f is strictly increasing,

$$\mathcal{S}_f(m \oplus n) = f(\mu(m \oplus n)) > f(\mu(m)) = \mathcal{S}_f(m),$$

and similarly $> \mathcal{S}_f(n)$.

Exercise

Exercise 4 (Landauer bound as a theorem schema). Let Len be a description length on histories and define $\Delta_\theta(m)$ as erased bits under $\mathcal{F}_\theta$. Assuming physical erasure semantics, derive $Q \geq k_B T \ln 2 \cdot \Delta_\theta(m)$. State clearly what is an assumption and what is derivation.

Worked Example / Guided Work

Assumption: erasing one bit irreversibly at temperature T costs at least $k_B T \ln 2$ heat. If Δ bits are erased, linearity gives the bound.

Solution

Solution. Assume Landauer's principle: irreversible erasure of one bit costs $\geq k_B T \ln 2$. For $\Delta_\theta(m)$ erased bits, the minimum heat is

$$Q \geq (k_B T \ln 2) \cdot \Delta_\theta(m).$$

Derivation is linear aggregation; the physical postulate is the one-bit bound.

Exercise

Exercise 5 (Decay semigroup property in mass). Suppose a decay family satisfies $\mu(m \ominus_d \lambda) \leq \lambda\ \mu(m)$. Show:

$$\mu((m \ominus_d \lambda) \ominus_d \rho) \leq (\lambda\rho)\ \mu(m).$$

Worked Example / Guided Work

Apply the inequality twice:

$$\mu((m \ominus_d \lambda) \ominus_d \rho) \leq \rho\ \mu(m \ominus_d \lambda) \leq \rho(\lambda\, \mu(m)).$$

Solution

Solution. Directly:

$$\mu((m \ominus_d \lambda) \ominus_d \rho) \leq \rho\ \mu(m \ominus_d \lambda) \leq \rho(\lambda\ \mu(m)) = (\lambda\rho)\, \mu(m).$$

Exercise

Exercise 6 (Distributivity witness: show non-$\equiv_s$ generically). In the ordered-tree canonical model, argue that the two histories in the distributivity diagram are not isomorphic in general. Give a sufficient condition on the leaf labels a, b, c.

Worked Example / Guided Work

Root labels differ: left root is $\times$, right root is $+$. Isomorphisms preserve labels, so if $\times \neq +$ as labels, they cannot be isomorphic.

Solution

Solution. In the canonical model, labels are preserved under isomorphism. The left history root is $\times$ while the right history root is $+$. Since $\times \neq +$, they cannot be isomorphic. A sufficient condition is simply that operation labels are distinct and preserved.

Exercise

Exercise 7 (Entropy forbids structural time reversal). Assume strict second law: $\mathcal{S}(m \oplus n) > \mathcal{S}(m)$ for all n. Show there is no operator R such that $\mathcal{S}(R(m \oplus n)) = \mathcal{S}(m)$ for all m, n unless R implements forgetting.

Worked Example / Guided Work

If R returns an object with entropy equal to m, it must reduce entropy from $m \oplus n$ back to m. But entropy strictly increased. A reduction requires a mass-reducing primitive—forgetting.

Solution

Solution. Since $\mathcal{S}(m \oplus n) > \mathcal{S}(m)$, any R with $\mathcal{S}(R(m \oplus n)) = \mathcal{S}(m)$ must reduce entropy. Entropy reduction implies mass reduction for monotone entropy, which cannot occur in the forward language. Therefore R must incorporate forgetting (explicit erasure/compression) or cannot exist.

Exercise

Exercise 8 (Concrete mass computations). In the tree model where μ counts internal nodes, let

$$m = ((\mathbf{1} \oplus \mathbf{1}) \oplus \mathbf{1}), \qquad n = (\mathbf{1} \oplus (\mathbf{1} \oplus \mathbf{1})).$$

Compute $\mu(m)$ and $\mu(n)$ and verify that $\mu(m) = \mu(n)$ even though $m \not\equiv_s n$.

Worked Example / Guided Work

Both are binary trees with 3 leaves. Any full binary tree with 3 leaves has 2 internal nodes. Structural nonassociativity comes from different shape/order, not different internal-node count.

Solution

Solution. Each expression has 3 genesis leaves and 2 internal + nodes. So $\mu(m) = 2$ and $\mu(n) = 2$. The histories are non-isomorphic as ordered trees, hence $m \not\equiv_s n$ despite equal mass.

Exercise

Exercise 9 (A worked "irreversibility paradox" resolution). Consider two computations that produce the same value:

$$x = (a \oplus b) \oplus c, \qquad y = a \oplus (b \oplus c).$$

Explain why it is correct that there is no structural inverse mapping $x \mapsto y$ within the forward language, and explain how a forgetting policy could make them equivalent.

Worked Example / Guided Work

Forward language cannot change the already-built computation graph without introducing more joins. Rewriting associativity is not "free"; it is a structural transformation. A forgetting policy that discards bracket structure (e.g. quotient by child-permutation or by shape) can identify them, but that is explicit information loss.

Solution

Solution. Within $\mathcal{L}_{+,\times}$, any attempt to map x to y must construct a new history from x, which increases mass rather than replacing structure. Thus no structural inverse exists. If one applies a forgetting operator that prunes away internal association structure (e.g. truncation to depth 1 or quotient that forgets parenthesization), then $\mathcal{F}(x) \equiv_s \mathcal{F}(y)$ can hold: equivalence is purchased by explicit abstraction.

Exercise

Exercise 10 (Design an entropy that matches MDL). Suppose $\mathrm{Len}(h)$ is description length in bits. Propose an entropy functional $\mathcal{S}_{\mathrm{MDL}}(m)$ using $\mathrm{Len}(\mathrm{hist}(m))$ and show it is strictly increasing under strict growth if Len is strictly increasing in history size.

Worked Example / Guided Work

Candidate:

$$\mathcal{S}_{\mathrm{MDL}}(m) = \mathrm{Len}(\mathrm{hist}(m)) \quad \text{or} \quad \log(1 + \mathrm{Len}(\mathrm{hist}(m))).$$

If strict growth implies history size increases and Len increases with size, then entropy rises.

Solution

Solution. Define $\mathcal{S}_{\mathrm{MDL}}(m) = \log(1 + \mathrm{Len}(\mathrm{hist}(m)))$. If strict growth guarantees $\mathrm{hist}(m \oplus n)$ has strictly larger size than $\mathrm{hist}(m)$ and Len is strictly increasing with that size, then $\mathrm{Len}(\mathrm{hist}(m \oplus n)) > \mathrm{Len}(\mathrm{hist}(m))$ and monotonicity of $\log(1 + x)$ gives $\mathcal{S}_{\mathrm{MDL}}(m \oplus n) > \mathcal{S}_{\mathrm{MDL}}(m)$.

Chapter Summary

- Strict growth under composition creates an internal arrow of time.
- No-cancellation: structural inverses cannot exist without explicit forgetting.
- No-cycle: forward constructors cannot return to the same structural state.
- Entropy as any monotone function of mass yields a strict Second Law.
- Forgetting is the only mass-reducing primitive; compression must be explicit.
- Landauer semantics connect erased description length to physical dissipation.
- Decay families formalize controlled abstraction schedules over time.
- Classical identities survive at $\equiv_v$ but split into distinct computation graphs at $\equiv_s$.

Roadmap (Where This Goes Next)

Chapter 4 (next) should:

- Develop metric geometry deeply: explicit edit scripts, geodesics, curvature invariants.
- Define fragility and phase transitions rigorously; prove balanced-tree advantages under pruning.
- Extend from trees to DAG histories: unfoldings, quotients, and complexity classes.
- Provide larger "worked lab" examples: random tree ensembles, simulation hypotheses, and experiment design.
- Introduce categorical and order-theoretic structure: embedding preorders, cores, thin categories, limits.