Settlement-Based Accountability

A Witness-Verifiable Ordering Framework

Abstract

In many contemporary systems—distributed protocols, audit logs, AI decision pipelines, and legal or financial infrastructures—causality is routinely inferred from propagation order or physical time. However, such inference fails whenever observation is partial, propagation is asynchronous, or responsibility must be established independently of physical reachability.

This work introduces a formal framework in which ordering arises from **settlement** rather than propagation. A settlement is an irreversible, decision-gated state transition that must produce a publicly verifiable **witness**. The ordering induced by these witnesses defines a responsibility-bearing structure, independent of physical signal timing.

We present a minimal object language, explicit bridge assumptions to implementation, and a set of falsifiable claims. The framework is formalised in Coq, admits concrete instances, and makes its failure conditions explicit. This work does not aim to redefine physical causation; instead, it defines an auditable ordering suitable for systems where accountability, not influence, is the primary concern.

Part I · Motivation & Scope

1. The Problem of Accountability Ordering

1.1 When propagation order fails

Propagation order answers the question *"what could have influenced what"*.
It does not answer *"what is acknowledged as having happened"*.

In distributed systems, multiple events may propagate in different orders to different observers. In audit systems, records may exist locally but not be recognised globally. In AI decision pipelines, intermediate computations may occur without being attributable or reviewable.

In all these cases, propagation order is observable but **insufficient**. What matters is not when a signal arrived, but when a system **committed** to a state change.

1.2 Why responsibility cannot be read from time

Responsibility presupposes three properties:

1. **Acknowledgement** – the system recognises an event.
2. **Irreversibility** – the recognition cannot be silently undone.
3. **Attribution** – the recognition can be demonstrated to third parties.

Physical time alone provides none of these. A signal can arrive without being accepted. A computation can occur without being recorded. A state can change transiently without leaving a trace.

Responsibility therefore cannot be derived from temporal precedence. It requires an explicit notion of settlement.

1.3 Auditability as a first-class requirement

Modern systems are increasingly required to justify decisions *after the fact*:

- financial transactions,
- safety-critical automation,
- AI-assisted judgments,
- regulatory compliance.

In such contexts, an event that cannot be audited is functionally equivalent to an event that never occurred. Auditability is not an add-on; it is a structural requirement.

This motivates an ordering notion grounded in **what is written into history**, not what merely occurred in computation or propagation.

2. What This Work Is *Not*

2.1 Not a physical causality theory

This framework does not describe how physical influence propagates, nor does it impose constraints on spacetime, signals, or energy transfer.

Questions of light cones, simultaneity, and relativistic causation are explicitly out of scope.

2.2 Not a replacement for relativity

Physical theories already provide correct and necessary accounts of propagation limits. Nothing in this work contradicts or modifies those results.

The ordering defined here operates at a different level: it concerns **recognition and record**, not physical interaction.

2.3 Not an interpretation of quantum mechanics

This framework does not attempt to explain quantum correlations, measurement collapse, or non-local statistics.

While quantum systems may produce records or measurements that enter settlement-based systems, the physical interpretation of those phenomena lies outside the scope of this work.

3. Scope and Applicability

3.1 Systems with explicit decision and settlement

The framework applies to systems that satisfy the following structural properties:

- inputs are evaluated by a **decision procedure**;
- accepted inputs trigger **irreversible state transitions**;
- such transitions produce **verifiable evidence**;
- multiple observers may reconstruct ordering from evidence.

Examples include transaction commit protocols, blockchains, append-only logs, and formal audit trails.

3.2 Systems without settlement (out of scope)

Purely physical systems without explicit acknowledgment, reversible computations, or ephemeral internal states fall outside the intended domain.

In such systems, there is no principled notion of responsibility to order.

3.3 Summary of applicability boundaries

This framework is:

- **applicable** where accountability is required,
- **neutral** with respect to physical implementation,
- **explicitly limited** to systems that admit settlement and witness.

It makes no claim of universality beyond these boundaries.

Part II · Formal Core (Object Language)

4. Core Entities

This section introduces the minimal object language of the framework.
All definitions in this part are **internal**: they do not reference physical implementation, observers, or execution environments.

4.1 States and Inputs

Let S denote a set of **states**, representing the complete internal configuration of a system at a given moment.

Let Σ denote a set of **inputs**, representing externally supplied propositions, requests, or events submitted to the system.

No algebraic structure is assumed on either S or Σ.
In particular, states need not be ordered, measurable, or comparable.

4.2 Decisions

A **decision** is the system's judgment on an input, evaluated in a given state.

We define a three-valued decision space:

$$\text{Decision} \in \{acc, rej, \bot\}$$

where:

- acc denotes acceptance,
- rej denotes explicit rejection,
- $\bot$ denotes undefined or non-admissible evaluation.

A **decision function** is a mapping:

$$D: S \times \Sigma \to \text{Decision}$$

The value $\bot$ is semantically distinct from rejection: it represents the absence of a valid decision context, not a negative judgment.

4.3 Settlements

A **settlement** is an irreversible state transition that may occur only after acceptance.

We define a (partial) settlement function:

$$K: S \times \Sigma \to \mathrm{Option}(S)$$

where:

- $K(s, x) = \mathrm{Some}(s')$indicates a successful settlement from $s$to s',
- $K(s, x) = \mathrm{None}$indicates no settlement.

No assumptions are made about determinism beyond this interface.

5. Decision-Gated Settlement

The core invariant of the framework is that **settlement is gated by decision**.

5.1 Gating condition

Settlement is permitted **if and only if** the decision is acceptance:

- If $D(s, x) = acc$, then settlement may occur.
- If $D(s, x) \in \{rej, \bot\}$, then settlement must not occur.

Formally:

- Acceptance implies the existence of a post-state.
- Rejection or undefined evaluation implies the absence of settlement.

This condition prevents systems from silently transitioning state without explicit acknowledgment.

5.2 Events as settled decisions

We define an **event** as the successful settlement of an input:

$$\mathrm{Event}(s, x) \quad \Leftrightarrow def \quad \exists s'.\ K(s, x) = \mathrm{Some}(s')$$

Under this definition:

- A decision without settlement is not an event.
- A propagated signal without settlement is not an event.

- An internal computation without settlement is not an event.

Only what is settled counts as having occurred.

5.3 Absence of event without acceptance

A direct consequence of decision-gated settlement is the following:

If an input is not accepted, then no event occurs.

This property ensures that the event space of the system is exactly the image of accepted and settled inputs, and not a superset inferred from observation or execution traces.

6. Witnesses

Settlement alone is insufficient for accountability.
A settlement must leave behind **evidence**.

6.1 Witness generation

A **witness** is an artefact produced by a settlement, intended to be consumed by third parties.

Let W denote the witness type.

We define a witness extraction function:

$$\mathcal{W}: S \times S \to \mathrm{Option}(W)$$

such that a successful settlement from s to s' yields a witness w.

6.2 Public verification

Each witness must be **publicly verifiable**, without access to hidden state.

We define a verification predicate:

$$\mathrm{Verify}: W \to \{true, false\}$$

Verification is required to be:

- deterministic,

- state-independent,
- reproducible by any third party.

This condition prevents private or authority-dependent validation of settlement.

6.3 Completeness and soundness conditions

Witnesses are subject to two structural conditions:

- **Completeness**: every settlement yields a verifiable witness.
- **Soundness**: every verifiable witness corresponds to a settlement, possibly under additional bridge conditions.

These conditions connect internal state transitions to externally checkable evidence, without equating the two.

7. Settlement-Induced Ordering

The final component of the object language is an ordering relation derived from witnesses.

7.1 Witness-induced edges

Let $\prec \subseteq W \times W$ be a binary relation indicating that one witness precedes another.

This relation is **not** assumed to be total, nor derived from time.
Its interpretation is purely structural.

7.2 Transitive closure

We define the induced ordering $\prec^*$ as the transitive closure of $\prec$.

This closure represents indirect responsibility chains: if one settlement depends on another, the corresponding witnesses are ordered.

7.3 Acyclicity

A central invariant of accountability is **acyclicity**:

No witness may precede itself in the induced order.

Formally:

$$\forall w \in W,\ \neg(w \prec^{*} w)$$

This excludes circular responsibility and self-justifying records.

Part IV · Bridge Assumptions (Implementation Layer)

8. Monotone Resources and Irreversibility

The object language defines settlement as irreversible by fiat.
In any real system, irreversibility is never free.

This section introduces **explicit bridge assumptions** that connect formal settlement to implementation constraints.

8.1 Abstract resource model

We assume the existence of an abstract **resource domain** R, together with a strict partial order $\prec_R$.

A resource assignment function maps states to resource values:

$$\rho: S \to R$$

No concrete interpretation is imposed.
The resource may represent energy expenditure, entropy increase, ledger height, write-once counters, or any other monotone quantity provided by the implementation.

8.2 Strict monotonicity

The defining bridge assumption of irreversibility is:

Every settlement strictly increases resource.

Formally:

$$\forall s, s'.\ \mathrm{Settled}(s, s')\ \Rightarrow\ \rho(s) \prec_R \rho(s')$$

This assumption makes explicit that irreversibility is not a logical consequence of settlement, but a **cost paid by the implementation**.

8.3 Why irreversibility is not free

Without a monotone resource:

- settlements could be silently undone,
- witnesses could be replayed,
- responsibility chains could be rewritten retroactively.

By separating irreversibility into a bridge assumption, the framework avoids claiming what it cannot justify: irreversibility is enforced by consuming something that cannot be recovered.

9. Conflict and No-Double-Settlement

Settlement-based accountability requires that mutually exclusive claims cannot both be settled.

9.1 Conflict relations

We assume a **conflict relation**:

$$\mathrm{conflict} \subseteq W \times W$$

Two witnesses are in conflict if they assert incompatible settlements.
The exact meaning of incompatibility is system-dependent and deliberately left abstract.

9.2 Mutual exclusion of settlements

The core bridge assumption is:

Conflicting witnesses cannot both verify.

Formally:

$$\mathrm{conflict}(w_1, w_2) \;\Rightarrow\; \neg(\mathrm{Verify}(w_1) \wedge \mathrm{Verify}(w_2))$$

This is not a logical theorem.
It is a requirement imposed on any implementation that claims to support accountability.

9.3 What breaks if this fails

If conflicting witnesses can both verify:

- responsibility becomes ambiguous,
- settlement order loses meaning,
- auditability collapses.

Rather than hiding this failure mode, the framework elevates it to a **first-class falsification condition**.

10. Observer Model and Eventual Convergence

Accountability is meaningless if ordering depends permanently on the observer.

10.1 Partial observation

We assume a set of observers O, each of which may see only a subset of witnesses at any given time.

Observers may initially disagree on ordering.

This disagreement is not a bug; it reflects real-world asynchrony.

10.2 Honest observers

We distinguish **honest observers**: those who apply the verification and ordering rules correctly, without fabricating witnesses.

No synchrony or trust beyond rule adherence is assumed.

10.3 Eventual consistency of order

If a system claims **finality**, it must satisfy the following bridge assumption:

After some point, all honest observers converge on the same induced ordering.

Formally:

There exists a time after which any two honest observers, given sufficient witness exposure, derive identical ordering relations.

This assumption is optional but explicit.
Systems that do not satisfy it must not claim final accountability.

Part V · Falsifiability

This framework does not claim universality.
Instead, it defines **precise conditions under which it must fail**.

A system that violates any of the claims in this section does not "approximately" satisfy the framework; it **refutes** it for that domain.

11. Falsifiable Claims

All falsifiable claims are conditional on the bridge assumptions stated in Part IV. They are not theorems of the object language; they are **testable commitments** made by any implementation that adopts this framework.

11.1 Cycle construction (F1)

Claim (F1)
In any system satisfying the witness and ordering assumptions, the settlement-induced order is acyclic.

Formally:

There does not exist a witness w such that
$w \prec^* w$.

How to falsify
Produce a finite set of witnesses $\{w_1, \ldots, w_n\}$ such that:

$$w_1 \prec w_2 \prec \cdots \prec w_n \prec w_1$$

and each w_i verifies successfully.

Such a construction demonstrates circular responsibility and invalidates the framework for that system.

11.2 Double-settlement attacks (F2)

Claim (F2)
Conflicting settlements cannot both verify.

Formally:

If $\mathsf{conflict}(w_1, w_2)$, then
$\neg(\mathsf{Verify}(w_1) \wedge \mathsf{Verify}(w_2))$.

How to falsify
Produce two witnesses that:

1. assert incompatible settlements,
2. are declared conflicting by the system's own rules,
3. both pass public verification.

If such witnesses exist, then the system admits contradictory histories and cannot support accountability ordering.

11.3 Environment-invariant ordering (F3′)

Claim (F3′)
If ordering rules do not depend on propagation parameters, then changing the environment must not change the induced order.

This claim is explicitly conditional.

If an implementation asserts that its ordering rules are independent of latency, bandwidth, or observation delay, then the derived ordering must be invariant under changes to those parameters.

How to falsify
Fix the rules and witness set.
Vary only environmental parameters (e.g. message delay, observation timing).

If two honest observers, applying identical rules to the same witnesses, derive incompatible orderings solely due to environmental variation, then the claim is false for that system.

12. How to Break the Framework

This framework invites attack.
Not rhetorically, but constructively.

12.1 Required counterexamples

A valid refutation must include:

- concrete witnesses,
- a publicly verifiable verification procedure,
- a demonstrable violation of F1, F2, or F3′.

Narrative arguments, informal intuition, or appeals to implementation difficulty do not constitute refutation.

12.2 What counts as a valid refutation

A refutation is valid if:

- it respects the stated bridge assumptions,
- it operates within the declared scope,
- it produces verifiable artefacts.

Refutations that rely on violating assumptions outside scope (e.g. denying irreversibility while claiming accountability) are not considered meaningful.

12.3 Limits of formal guarantees

Formalisation guarantees internal consistency, not empirical adequacy.

This framework does not claim that all real systems satisfy its assumptions.
It claims only that:

Any system that claims settlement-based accountability must either satisfy these conditions or accept refutation.

Part VI · Concrete Instances

This section presents concrete systems that instantiate the framework.
The goal is not realism or completeness, but **demonstrable satisfiability**.

Each instance answers a single question:

Does there exist a nontrivial system in which decision-gated settlement, verifiable witnesses, and accountable ordering can all be satisfied simultaneously?

13. Toy Model: Existence Proof

13.1 Purpose of the toy model

The toy model serves one purpose only: to demonstrate **model existence**.

It is deliberately minimal and unrealistic.
Its value lies not in applicability, but in showing that the abstract axioms are not mutually contradictory.

13.2 Structure of the model

- **State**: a natural number representing history length
- **Input**: a unit value (no semantic content)
- **Decision**: always accept
- **Settlement**: increment the state
- **Witness**: the new state value
- **Verification**: always succeeds
- **Resource**: state value (strictly increasing)

In this model:

- every accepted input settles,
- every settlement increases resource,
- the induced order is strictly acyclic.

13.3 What the toy model proves—and what it does not

This model proves:

- the framework admits at least one concrete instance,
- irreversibility and ordering can coexist without contradiction.

It does **not** prove:

- usefulness,
- robustness,
- resistance to adversarial behavior.

The toy model closes a logical gap, nothing more.

14. Transaction Commit System

The second instance is intentionally closer to real systems, while remaining formally tractable.

14.1 System overview

This instance models a simplified transaction commit mechanism.

- **State**: a ledger of committed transaction identifiers
- **Input**: a transaction identifier
- **Decision**: accept if and only if the identifier is not already present
- **Settlement**: append the identifier to the ledger
- **Witness**: a canonical commitment record
- **Resource**: ledger length

This captures a common pattern across databases, blockchains, and audit logs.

14.2 Decision and settlement

The decision procedure enforces **freshness**:
a transaction may be settled at most once.

Settlement appends the transaction to the ledger, creating an irreversible record.

No rollback mechanism is provided.

14.3 Witness design

Each settlement produces a witness containing:

- the transaction identifier,
- a canonical index binding it to the settlement.

Verification is public and deterministic:
any third party can confirm that the witness is well-formed.

14.4 Conflict and no double settlement

Conflicts are defined as witnesses asserting incompatible settlements for the same transaction.

The system enforces the following invariant:

Two conflicting witnesses cannot both verify.

This ensures that responsibility cannot be duplicated or forked.

14.5 Resource monotonicity

The ledger length serves as a monotone resource.

Each settlement strictly increases the resource value, preventing silent reversion.

14.6 Induced ordering

The ordering relation is derived from witness structure.

The transitive closure of this relation is acyclic, yielding a well-defined responsibility chain.

14.7 Limitations of the instance

This instance deliberately omits:

- concurrency control,
- fault tolerance,
- network behavior.

Those concerns belong to implementation-level analysis, not to the object language.

15. Summary of Instances

The instances demonstrate three key points:

1. The framework is internally consistent.
2. It admits nontrivial, verifiable models.
3. Its assumptions correspond to recognizable system design choices.

They do **not** claim universality or optimality.

Part VII · Discussion

16. Relation to Existing Work

This section situates the framework relative to established concepts, without attempting to subsume them.

16.1 Lamport clocks and logical time

Lamport clocks define a partial order over events in distributed systems, based on message causality.

The present framework shares a structural similarity: both define ordering without reference to physical time. However, the motivation differs:

- Lamport clocks answer *“what must have happened before what”*.
- Settlement-based ordering answers *“what is acknowledged and therefore accountable”*.

Logical clocks track causality of execution; settlement-based ordering tracks **commitment to history**.

16.2 Vector clocks and causal histories

Vector clocks refine Lamport clocks by encoding causal dependencies more precisely.

They describe *information flow*, not *recognition*.

A vector clock may record that an event occurred, even if the system later rejects or ignores it. In contrast, settlement-based ordering excludes all unacknowledged activity by definition.

This distinction matters whenever systems must justify outcomes to external parties.

16.3 Blockchain finality models

Blockchain systems already distinguish between:

- propagation (gossip),
- tentative inclusion,
- final settlement.

Finality mechanisms implicitly implement settlement-based accountability, often without explicit formalisation.

This framework makes those assumptions explicit and testable, rather than protocol-specific.

17. Accountability vs. Causality

17.1 Why this is a higher-layer abstraction

Physical causality constrains what can influence what.

Accountability constrains what can be **claimed**, **audited**, and **enforced**.

The latter presupposes the former but does not reduce to it.

Attempting to derive accountability solely from physical causation conflates influence with responsibility.

17.2 Multiple notions of "cause"

The term *cause* is overloaded across domains:

- physics: influence constrained by spacetime,
- logic: derivability,
- law: attribution of responsibility,
- computation: dependency.

This framework defines a notion aligned with the last two. It does not claim exclusivity.

17.3 Why physical causation is orthogonal

A system may respect physical causation perfectly while failing accountability:

- signals propagate correctly,
- computations execute as expected,
- yet no authoritative record exists.

Conversely, a system may enforce accountability ordering without adding any new physical constraints.

These concerns operate on orthogonal axes.

18. Design Trade-offs

18.1 State-free vs. state-dependent verification

State-free verification maximises universality and third-party auditability but requires additional bridge assumptions (e.g. freshness).

State-dependent verification simplifies soundness but narrows applicability.

The framework accommodates both, provided assumptions are explicit.

18.2 Generality vs. enforceability

More general definitions admit more systems but require stronger bridge assumptions.

Stricter definitions simplify enforcement but limit scope.

The framework does not resolve this tension; it exposes it.

18.3 What the framework refuses to do

The framework refuses to:

- infer accountability from execution traces alone,
- hide irreversibility behind abstraction,
- claim validity outside declared assumptions.

These refusals are intentional.

Part VIII · Conclusion (Non-Teleological)

19. What Has Been Defined

This work defines:

- a minimal object language for decision-gated settlement,
- an explicit witness mechanism,

- a responsibility-bearing ordering induced by verifiable artefacts,
- a set of bridge assumptions that connect formal structure to implementation,
- precise conditions under which the framework must fail.

Nothing more.

20. What Is Left to the World

This framework does not assert that real systems satisfy its assumptions.

It asserts that:

Any system that claims accountability must either satisfy these conditions or accept refutation.

Whether the framework is adopted, modified, or rejected is an empirical matter.

21. Final Statement

This work does not seek to explain how the world operates.

It seeks to formalise how systems **commit to history**.

If the formalism is inadequate, counterexamples will emerge.
If it is adequate, it will be used.

Either outcome completes the work.

Kaifanxie_20260131