0.1 Rethinking Atomicity: Counterfactual Transactions

From ./AE-Specifications-ETH/standalone/Transactions-maybe-dup.tex

This document challenges the Forward-In-Time-Only (FITO) assumptions behind conventional transactions in distributed systems. It argues that atomicity, as currently conceived, is a flawed abstraction and proposes a framework for reversible subtransactions as a more robust alternative.

> *"Transactions begin and they end."*
> —Charlie Johnson, TMF Product News

This simple phrase conceals deep design hazards. Transactions appear to begin with a trigger and end with a commit, but in distributed systems, these bookends obscure severe internal inconsistencies.

At issue are the mechanisms we use to track and guarantee these transactional intervals: timestamps, logs, filesystems, and even our concepts of causality. Each introduces cracks in the facade of atomicity.

0.2 The Forward-In-Time-Only Fallacy

FITO: Forward-In-Time-Only thinking assumes linear causality.

Most distributed systems today adopt what we call **Forward-In-Time-Only (FITO)** thinking. That is:

1. **Open a transaction** with a timestamp.
2. **Apply a sequence** of operations.
3. **Close the transaction** with a commit or rollback.

But this approach breaks down under scrutiny.

Three FITO Hazards

1. **Timestamps are not unique.** Even on a single machine with GHz processors and nanosecond clocks, timestamp collisions occur. OS-level clock management does not guarantee uniqueness.
2. **Timestamps are single points of failure.** Any drift, packet loss, or sync error in NTP/PTP introduces false ordering assumptions.
3. **Simultaneity is an illusion.** Relativity tells us simultaneity is observer-dependent. Building global event orderings on timestamps is unsafe.

0.3 The False Comfort of Atomicity

We often say: "all or nothing." But our stack is built on sand:

- The database relies on the log.
- The log relies on the filesystem.
- The filesystem relies on `fsync`.
- `fsync` relies on storage hardware.

Each of these layers fails to guarantee true atomicity. When one fails, the recovery model becomes: **Smash and Restart**.

0.4 The Myth of Reliable Commit

Protocols like Two-Phase Commit (2PC) attempt to enforce distributed agreement. But they depend on:

- Log synchronization across nodes
- Network reliability
- Time-based coordination

When any assumption breaks, so does safety. Eventually, we replace consistency with survivability—and correctness with heuristics.

0.5 Toward Reversible Thinking

Suppose we reject FITO. Suppose we view the transaction as *reversible*.

> If the forward protocol is correct, we can construct a reverse protocol.

This leads to **reversible subtransactions**: bounded operations that can be undone without global rollback.

Counterfactual Transactions

- A transaction can end, then begin again.
- Logs become braids, not linear sequences.
- Atomicity becomes a constraint, not an assumption.

Inspired by Marletto's counterfactual physics, this model embraces partial reversibility as an engineering practice.

0.6 Closing the Interval—Reopened

Simultaneity is not fundamental. Causality is.

"Closing the interval" with a commit only makes sense if we know the state is stable. In reality, it's a guess based on layers of non-atomic operations.

By rethinking transactions through reversible logic, we can:

- Define precise causal dependencies
- Undo partial effects
- Recover without restart

Reversibility isn't science fiction. It is what rollback always wanted to be.

0.7 Conclusion

The abstraction of atomicity has outlived its usefulness as a guarantee. In modern distributed systems, FITO thinking and timestamp dependency introduce hazards we can no longer ignore.

It is time to engineer **reversible protocols**, built on causal semantics—not illusions of simultaneity. Let us design transactions that don't just commit or roll back, but that can *unwind*.