Systemisation of Knowledge: Digital Liquidity

This document is a planning outline for the first of four papers in a PhD thesis on digital liquidity. It establishes the foundational framework—the Balance-State Transition Event (BSTE) model and Digital Liquidity Stack—that underpins the three subsequent papers, which are described in Section **??**.

1 Purpose and Positioning

1.1 High-Level Aims

A unified, event-centric framework for *digital liquidity* across both legacy and tokenised infrastructures.

- Provide a **minimal, machine-checkable event model** (Balance-State Transition Event (BSTE)) for balance-state transitions.
- Define a **Digital Liquidity Stack** that captures money nature, ledger technology, clearing, settlement, finality, credit, encumbrance, and interoperability.
- Use this to systematically classify payment systems, Financial Market Infrastructures (FMIs), and tokenised systems (Real-Time Gross Settlement (RTGS), Automated Clearing House (ACH) / Deferred Net Settlement (DNS), card schemes, Continuous Linked Settlement (CLS), Central Counterparty (CCP) cash, stablecoins, Central Bank Digital Currencies (CBDCs), bridges, Automated Market Makers (AMMs), unified ledgers).
- Surface **patterns and failure modes** in digital liquidity management.

1.2 Meaning of "Digital Liquidity"

Liqudity has multiple meanings:

(a) **Market liquidity**: order-book depth, bid–ask spreads, price impact, AMM slippage.
(b) **Funding and settlement liquidity**: ability of institutions and infrastructures to obtain and deploy cash/collateral to meet obligations on time.

In this work:

> **Digital liquidity** means the *capacity of digital infrastructures and participants to execute balance-state transitions* (our BSTEs)—to make obligations good in the right asset, on the right ledger, at the right time.

We *explicitly* do **not** attempt to survey or systematise market microstructure (order books, AMM curve design, price impact). Those questions are treated as upper-layer phenomena and will be the subject of subsequent work.

1.3 Intended Contributions

The chapter / article delivers:

(C1) **BSTE model**: a minimal, compositional event representation for digital value movements.
(C2) **Digital Liquidity Stack**: a 10-dimension core classification plus extended attributes to locate any system in a design space.
(C3) **Mechanism taxonomy**: a systematic description of holds, locks, collateralisation, credit, queues, netting, Liquidity Savings Mechanisms (LSM), Payment versus Payment (PvP) / Delivery versus Payment (DvP) / Payment on Payment (PoP), channels, and bridges as compositions of BSTEs.
(C4) **Comparative mapping**: worked classifications of representative legacy rails and tokenised systems.
(C5) **Research agenda**: a bridge from this plumbing-level Systematization of Knowledge (SoK) to (i) formal optimisation (Paper 2) and (ii) empirical market stylised facts (Paper 3).

2 Scope of Background Review

The following topics are in scope:

- Classical payment systems and FMI literature:
  - RTGS design, LSM and gridlock-resolution algorithms.
  - ACH/DNS systems, netting, risk management.
  - CCPs and CLS (PvP systems), liquidity implications.
- CBDC and unified-ledger architecture papers.
- Stablecoins, tokenised deposits, and tokenised collateral networks.
- Decentralized Finance (DeFi) systems: AMMs, lending protocols, cross-chain bridges.
- Existing "stacks" or taxonomies (e.g., generic blockchain stacks, CBDC design taxonomies).
- SoK methodology references (how SoK papers are typically structured).

The focus is **not** on exhaustive survey, but on situating this work among:

- payment/FMI engineering,
- tokenisation / Distributed Ledger Technology (DLT) infrastructure,
- SoK literature on distributed systems and crypto.

3 BSTE: Balance-State Transition Event

3.1 Economic Owner Abstraction

Introduce a conceptual mapping:

$$\text{econ\_owner}(\mathit{account}) \to \text{beneficial owner (bank, customer, CCP, etc.)}.$$

This allows us to distinguish:

- movements that change who owns the claim,
- movements that only change how an owner's claim is encumbered.

3.2 Primitive Event Types

Primitive P1: OWNERSHIP_TRANSFER

- Economic owner multiset changes.
- Supply S on the ledger is unchanged.
- Examples:
  - RTGS credit from bank A to bank B.
  - On-chain Ethereum Request for Comments 20 (ERC20) token transfer.
  - AMM swap legs where users receive tokens and AMM pool balances change.

Primitive P2: ENCUMBRANCE_ADJUST

- Economic owner set *unchanged*, but claims move between:
  - free balance (`available`),
  - encumbered buckets (holds, collateral, escrow, channels, etc.).
- Constraint: $\text{econ\_owner}(\text{src\_account}) = \text{econ\_owner}(\text{dst\_account})$.
- Examples:
  - Card pre-authorisation: available $\to$ reservation.
  - Central bank collateralisation: available $\to$ collateral.
  - Hashed Time-Locked Contract (HTLC) lock: available $\to$ channel or bridge_lock.
  - Release of a hold: reservation $\to$ available.

Primitive P3: SUPPLY_ADJUST

- Net change in recognised supply S.
- Exactly one of `src_account`, `dst_account` equals `EXTERNAL_SOURCE` or `EXTERNAL_SINK`.
- Examples:

– Central bank monetary operations in reserves.
– Stablecoin mint/burn.
– Protocol base-fee burn.

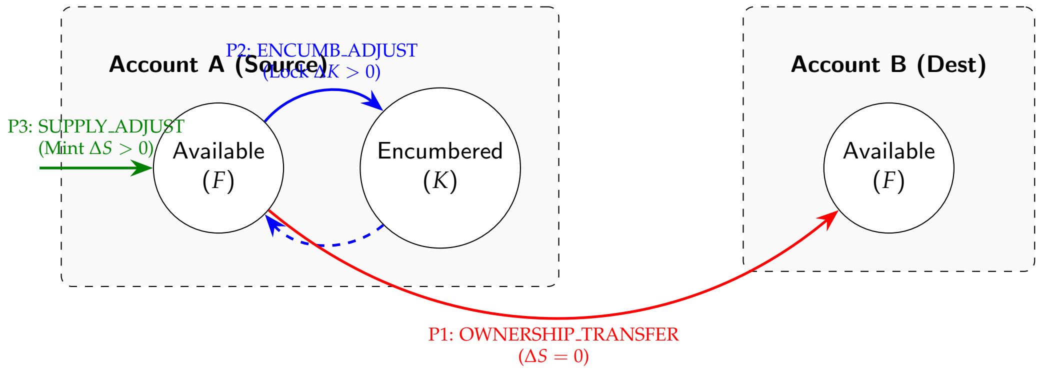


Figure 1: The Three Kernel Primitives of the BSTE Model. All digital value movement is composed of these atomic transitions between Free (F) and Encumbered (K) sub-balances, or between ledgers/the external system.

3.3 Canonical BSTE Schema

We adopt the following design choices:
- `amount` is strictly positive.
- No structural `NULL`; use explicit sentinels.
- Encumbrance is derived from bucket types; no `lock_delta`.

Identity and Ordering

1. `event_id` : unique identifier.
2. `ledger_id` : authoritative balance-record; may be Central Bank (CB) RTGS, bank core, scheme settlement file, or DLT state machine.
3. `asset_code` : identifies the asset.
4. `op_kind` : {OWNERSHIP_TRANSFER, ENCUMBRANCE_ADJUST, SUPPLY_ADJUST}.
5. `t_occurred` : logical posting time.
6. `event_seq` : per-ledger total order index.

Accounts and Balance Types

7. `src_account` : string, or `EXTERNAL_SOURCE`.
8. `src_balance_type` : {available, reservation, collateral, escrow, channel, internal_pending, bridge_lock, other}.
9. `dst_account` : string, or `EXTERNAL_SINK`.
10. `dst_balance_type` : same enum.

Encumbered buckets are those with `balance_type` $\neq$ `available` (but we may specify the exact list).

Supply and Economic Role

11. `supply_delta` : signed change to supply on the ledger.
12. `economic_role` : {principal, fee, tax, interest, margin, collateral_movement, other}.
13. `tax_subtype` : optional detail for `economic_role = tax`.

All fees/taxes are represented as ordinary BSTEs with appropriate `economic_role`.

Grouping and Atomic Sets

14. `group_id` : business group (Foreign Exchange (FX) trade, batch, clearing cycle).
15. `link_id` : groups events into an atomic or quasi-atomic set.

Atomic semantics are kept in a separate table `atomic_sets`, keyed by `link_id`, with fields:
- `atomic_pattern` : none, PvP, DvP, PoP.
- `atomic_mechanism` : single_ledger_tx, central_novation, htlc, escrow_agent, optimistic_with_fraud_proof, trusted_coordinator.
- `atomic_params` : JSON (timeout heights, hashlocks, etc.).
- `fx_rate` and `price_reference` : optional for cross-asset sets.

Evidence, Expiry, Notes

17. `message_ref` : upstream messages/logs (International Organization for Standardization (ISO) 20022 financial messages, ISO 8583 card transaction messages, transaction hashes).
18. `purpose_code` : business purpose.
19. `expiry_time` : for encumbrances with timeouts (HTLCs, auth holds, etc.).
20. `notes` : free text.

4 Pending Events

Introduce Pending-BSTE (Pending Balance-State Transition Event (PBSTE)) as an operational extension:
- Structural fields mirror BSTE.
- Additional fields:
  - `t_proposed`,
  - `status` $\in$ {pending, accepted, rejected, cancelled}.

PBSTEs do not affect:
- supply $S(t)$,
- encumbrance $K(t)$,
- free balances in the canonical historical accounting.

They do affect projected liquidity:

$$F_{\text{projected}}(t) = F_{\text{posted}}(t) - \sum_{\text{pending outflows}} \text{amount}.$$

5 Global Invariants and Bridging Semantics

5.1 Per-Ledger Supply and Encumbrance

For each (`ledger_id`, `asset_code`):
- Supply:
$$S(t) = S(t_0) + \sum_{\text{events } \leq t} \text{supply\_delta}.$$
- Encumbered quantity:
$$K(t) = \sum_{\text{accounts, encumbered buckets}} \text{balance}(t).$$
- Free balance per account:
$$\text{FreeBalance}(t) \geq 0.$$

5.2 Primitive-Level Constraints

- **P1 OWNERSHIP_TRANSFER**: `supply_delta = 0`.
- **P2 ENCUMBRANCE_ADJUST**: `supply_delta = 0` and econ_owner(src) = econ_owner(dst).
- **P3 SUPPLY_ADJUST**: exactly one endpoint is external.

5.3 Atomic Sets

For each `link_id`, there is an entry in `atomic_sets` describing the intended pattern and mechanism.
Invariants include:

- For PvP: no leg should settle in isolation (interpretation depends on mechanism).
- For DvP: cash and security legs are coupled.
- For HTLCs: encumbrances must be released by either success or timeout.

5.4 Bridging Invariants

For lock-and-mint bridges:

- Backing ledger uses `bridge_lock` buckets for locked units.
- Wrapped asset ledger mints new tokens via **SUPPLY_ADJUST**.

A simple invariant for one-to-one lock-mint:

$$S_{\text{wrapped}}(t) \leq K_{\text{backing,bridge\_lock}}(t)$$

(equality up to fees/slippage).

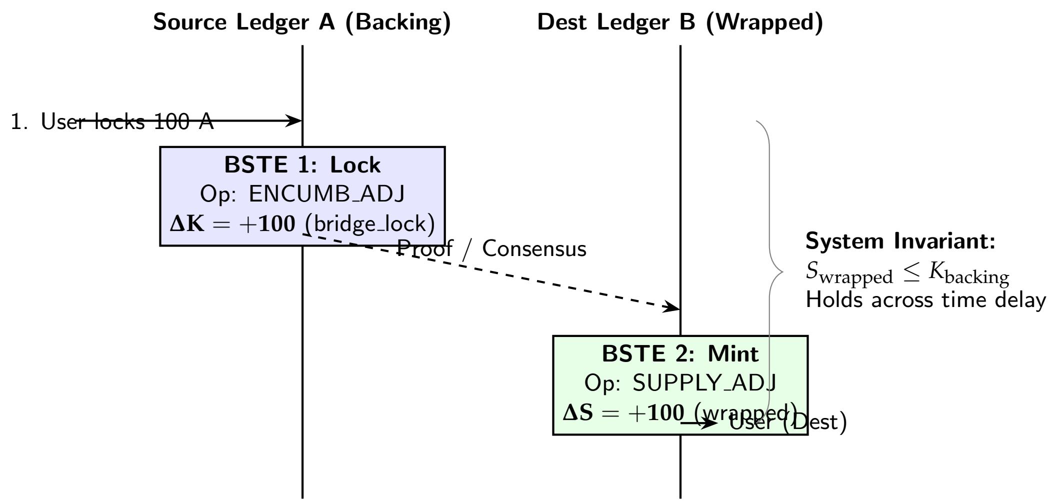


Figure 2: BSTE Sequence for Lock-and-Mint Bridging. The increase in S on the destination ledger is matched by an increase in K (Encumbrance) on the source ledger, ensuring the global free quantity remains conserved.

6 Digital Liquidity Stack (Core and Extended)

6.1 Core Dimensions

These are the 10 core fields used for classification tables.

D1. asset_nature: cb_reserve, cb_cash, commercial_deposit, e_money, stablecoin_fiat, stablecoin_crypto, token_native, security_cash, other.

D2. legal_form: balance_sheet_claim, trust_unit, fund_share, bearer_instrument, synthetic_derivative, cb_direct.

D3. representation_model: native_account, native_token, wrapped_mirror, synthetic.

D4. ledger_tech: cb_rtgs, bank_core, ccp_cash_ledger, cls_pvp, dlt_public, dlt_permissioned, scheme_internal, channel_state, other.

D5. account_model: account_balances, Unspent Transaction Output (utxo), smart_contract_state, hybrid.

D6. scheme_type: rtgs_operator, instant_payments, ach_dns, card_scheme, correspondent_network, Decentralized Exchange (dex)_protocol, bridge_protocol, mobile_money_scheme, other.

D7. access_model: direct, indirect, retail, wholesale, permissionless, permissioned.

D8. clearing_mechanism: none_gross, bilateral_net, multilateral_net, queue_lsm, continuous_net, offchain_channels.

D9. settlement_mode: gross, bilateral_net_batch, multilateral_net_batch, hybrid_queue_lsm, onchain_gross, onchain_batch.

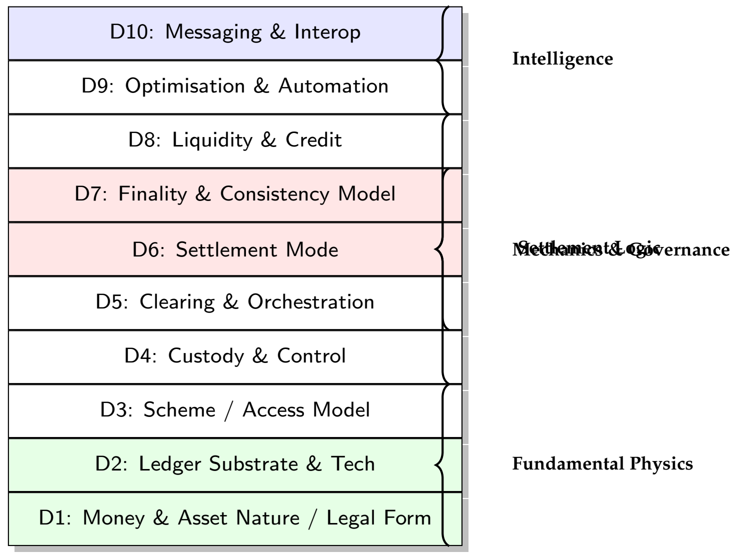


Figure 3: The Digital Liquidity Stack (10 Core Dimensions). Each system/asset is located by a coordinate in this design space.

D10. **finality_kind** and **consistency_model**: deterministic / deterministic_deferred / probabilistic vs eventual_reconciliation / consensus_atomic / hybrid.

6.2 Extended Dimensions

Extended fields that can be used in deeper analysis or thesis-only tables:

- custody_model, authorisation_model, freeze_authority.
- credit_sources, encumbrance_eligibility, allows_negative_balances.
- optimisation_style, optimisation_scope.
- messaging_standards, addressing_scheme, id_scheme.
- privacy_model, replay_guard.
- governance_model, legal_finality_basis.

7 Tokenised vs Non-Tokenised vs Hybrid

7.1 Non-Tokenised Systems

Common characteristics:

- representation_model = native_account,
- ledger_tech = cb_rtgs / bank_core / scheme_internal,
- consistency_model = eventual_reconciliation,
- execution is message-driven, core state is opaque during processing.

7.2 Tokenised Systems

Common characteristics:

- representation_model ∈ {native_token, wrapped_mirror},
- ledger_tech = dlt_public or dlt_permissioned,
- consistency_model = consensus_atomic,
- programmability: smart contracts see state and update in the same transaction.

7.3 Hybrid Systems

Examples:

- Tokenised deposits that sit atop bank cores but expose a token interface.
- RLN / unified-ledger designs with central-bank and commercial-bank tiers.
- Card schemes or Payment Service Providers (PSPs) that mirror balances onto a DLT sub-ledger.

Discussion will emphasise:
- how hybrid systems occupy intermediate coordinates in the stack,
- distinct failure modes and liquidity behaviours.

8 Taxonomy of Liquidity Mechanisms

This section classifies mechanisms as compositions of BSTEs at specific stack coordinates.

8.1 Credit and Funding

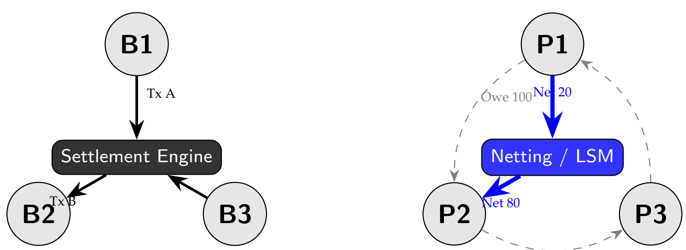
- Intraday central-bank credit (RTGS).
- Overdrafts and bilateral credit lines.
- Repo-based liquidity provision.

8.2 Encumbrances and Collateral

- Holds (card, instant payments).
- CCP margin and haircuts.
- Collateralisation at central bank / CCP / DLT-based vaults.

8.3 Clearing and Netting

- Bilateral and multilateral netting.
- LSM / gridlock resolution.
- DNS transfer cycles and settlement windows.



A. Gross Settlement (RTGS) **B. Net Settlement (ACH/LSM)**

Figure 4: Liquidity Mechanism Topologies. Gross Settlement (A) requires flow equal to the full transaction value. Net Settlement (B) resolves obligations (dashed) into smaller, net flows (solid) via an aggregator, conserving liquidity.

8.4 Channels and Off-Chain Mechanisms

- Payment channels (Lightning and analogues).
- State channels, rollups with periodic settlement.

8.5 PvP, DvP, PoP, and Bridges

- PvP FX legs across RTGS or DLTs.
- DvP securities settlement vs cash.
- PoP patterns where obligations are offset.
- Bridge designs: custodial lock-mint, burn-and-mint, synthetic representations.

9 Representative Case Studies

This section will contain 3–5 in-depth examples, each with:
- stack coordinates,
- BSTE sequences for typical flows,
- encumbrance and credit implications.

Candidate case studies:
1. **Central-bank RTGS with LSM**. Show queued payments, LSM runs, BSTE-level effect.
2. **Card scheme + DNS**. Auth holds, clearing batches, RTGS settlement, chargebacks.
3. **Instant retail system (e.g., New Payments Platform (NPP)-like)**. Real-time credits with holds and CB RTGS backing.
4. **DeFi AMM swap + cross-chain bridge**. Multi-leg on-chain bundles, HTLCs, finality and latency.
5. **Tokenised deposit / unified ledger**. Hybrid consistency, legal form, governance.

10 Design Space and Discussion

Here the chapter synthesises:
- How different systems cluster in the 10-dimensional core space.
- Trade-offs:
  - pre-funded vs credit-backed,
  - gross vs net settlement,
  - centralised vs consensus-based finality,
  - transparency vs privacy,
  - simplicity vs programmability.
- Failure modes:
  - reconciliation drift,
  - reorg risk,
  - stuck encumbrances (e.g., HTLC timeouts, unresolved holds),
  - bridge misconfigurations and backing failures.

This section also positions emerging designs (Regulated Liability Networks (RLN), unified ledgers, tokenised collateral networks) in the space.

11 Subsequent Papers Overview

11.1 Stylised Facts of Tokenised Real World Asset Markets (Paper 2)

Explain how:
- BSTE encoding allows uniform extraction of transaction and settlement data.
- Stack coordinates inform:
  - which systems are comparable,
  - where latencies and failures originate.
- Empirical stylised facts can then be tied explicitly to plumbing choices.

11.2 Collateral as Settlement Asset (Paper 3)

Show how:
- legal_form, governance_model, and encumbrance semantics in the stack become central to designing repo-native money and collateral tokens.
- invariants from the SoK constrain safe designs for yield-bearing settlement assets.

11.3 Agentic Liquidity (Paper 4)

Outline how the BSTE + stack model supports:
- Formal graph representations of multi-ledger liquidity.

- Problem statements of the form:
  "Given required OWNERSHIP_TRANSFER BSTEs, find ENCUMBRANCE_ADJUST and routing decisions that minimise a cost functional such as $\int K(t)\,dt$ subject to constraints."
- Integration of LSM/queueing, Mixed Integer Linear Programming (MILP) / Model Predictive Control (MPC), Reinforcement Learning (RL) agents.