SoK: Digital Liquidity

State-Machine and Type-System Ontology

Abstract

Digital liquidity today spans central-bank RTGS systems, ACH/card schemes, commercial-bank cores, super-app wallets, public blockchains, L2s, DeFi protocols, stablecoins, tokenised MMFs, and multi-CBDC platforms. Each comes with its own terminology and risk model. Systems researchers see "RTGS vs DeFi vs CBDC vs stablecoins" as qualitatively distinct systems, and lack a unified way to compare them, build tools over them, or formally reason about their behaviour.

This paper defines a **Systematisation of Knowledge (SoK)** for digital liquidity that addresses this fragmentation. Its core contributions are:

1. A **Balance-State Transition Event (BSTE)** model: A universal, event-level state-machine model for monetary ledgers, with three irreducible primitives: **OWNERSHIP_TRANSFER (P1)**, **ENCUMBRANCE_ADJUST (P2)**, and **SUPPLY_ADJUST (P3)**.
2. A **Core Digital Liquidity Profile (C1–C20)**: A finite feature space that classifies any instrument–infrastructure pair along asset, ledger, access, trust, privacy, ordering, and failure dimensions.
3. An explicit **alignment with Basel III, PFMI, and ISO 20022**, showing how existing standards map into this state-centric view.
4. A view of the SoK as a **state-machine type system** for smart contracts and protocols: given a contract's SoK type, we can derive obligations and verify them with formal methods.
5. A formal approach to **Liquidity Calculus**, providing mathematical definitions for digital LCR/NSFR and optimisation costs.
6. A comprehensive library of **20 case mappings** (Fedwire, ACH, Bitcoin, Tornado Cash, etc.) with the full 20-dimensional profile for each.

The result is a **computational ontology** and **front-end semantic type system for *digital liquidity instruments and infrastructures***, designed to plug into monitoring, optimisation, protocol design, and formal verification workflows.

Contents

1 Introduction

Digital liquidity infrastructure has exploded in diversity:

- Central-bank infrastructures (RTGS, wholesale CBDC pilots, multi-CBDC platforms).
- Commercial-bank rails (ACH, card schemes, instant payments, tokenised deposits).
- FMIs (CLS, CCP cash ledgers, CHESS-style settlement systems).
- Public-chain ecosystems (BTC, ETH, L2 rollups, DeFi AMMs, lending protocols).
- Tokenised cash/collateral networks (stablecoins, ERC-4626 vaults, tokenised MMFs).
- Privacy-preserving networks (Mixers, Shielded Pools, ZK-Rollups).

These are usually analysed in silos: an economist might compare RTGS and ACH; a DeFi researcher might analyse AMMs; an applied-crypto person might scrutinise bridges. There is no unified, technically precise **design space** for "digital liquidity" in which all of these systems can be represented, compared, and reasoned about.

From a systems perspective, there are three core problems:

1. **No common event model:** ISO 20022 and SWIFT provide rich messaging schemas, but they are *not* ledger state models. On-chain systems have tx logs, but semantics differ (account vs UTXO vs smart-contract state). We want a single event-level abstraction that covers RTGS, ACH, DeFi, and Lightning.
2. **No shared feature space:** We lack a finite set of dimensions that capture what matters for digital liquidity - not only asset nature, but privacy models, ordering mechanisms (MEV), and failure modes.
3. **No "type system" for contract behaviour:** Languages like Solidity or Rust tell you *how* a contract is implemented. They do not tell you *what kind of instrument or rail it is supposed to be* (e.g., fully-reserved stablecoin vs unsecured credit token) nor what invariants that implies.

This paper addresses these problems by defining a universal **Balance-State Transition Event (BSTE)** model (§3) and a **Core Digital Liquidity Profile** (C1–C20) (§4). Together, these constitute a **formal ontology** that acts as a semantic type system for digital finance.

2 Background and Requirements

2.1 Digital Liquidity and Instruments

We use "digital liquidity" as an umbrella term for the short-horizon, transactional layer of modern finance:

- monetary claims used as **settlement assets** (central-bank reserves, CBDCs, tokenised MMFs),
- **payment instruments** (bank deposits, e-money, stablecoins, platform balances),
- **collateral instruments** used in secured funding (repo, margin, RWA vault shares),
- and the infrastructures that move, encumber, and settle them.

2.2 State Machines vs Messaging Layers

At the highest level, a **ledger** is a **state machine**:

- **State**: for a given $(ledger\_id, asset\_code)$, balances per account (available, reservation, collateral, etc.) and summary variables.
- **Transitions**: events that update this state.

Different technologies give different encodings (Account vs UTXO vs ZK-Commitment). By contrast, systems like SWIFT + ISO 20022 are primarily **messaging networks**. They define messages, but not a canonical 'State'. Our SoK treats each ledger as a state machine with a BSTE log.

3 Balance-State Transition Event (BSTE) Model

3.1 Intuition

A **Balance-State Transition Event (BSTE)** is a single **atomic change** in balances of *one asset* on *one ledger*. If you replay all BSTEs in time order, you reconstruct the ledger's monetary state. This abstracts over Account-based, UTXO-based, and Zero-Knowledge (commitment-based) ledgers.

3.2 Unified Logical Schema

This schema merges the rigorous data requirements of ISO 20022 with the privacy and ordering primitives required for distributed ledger technology (DLT) and ZK systems.

```
struct BSTE:
    # --- Identity & Time ---
    event_id: string        # UUID, TxHash, or UETR
    ledger_id: string       # "CB.RTGS/US", "DLT/ETH", "BANK/DE"
    t_occurred: timestamp   # ISO 8601 / RFC3339

    # --- Ordering (for Distributed Systems) ---
    sequence_index: uint64  # Block height + Tx Index (Ordering)
    causal_ref: list[ID]    # Vector clock / dependency graph

    # --- The Primitive ---
    op_kind: enum { OWNERSHIP_TRANSFER, ENCUMBRANCE_ADJUST, SUPPLY_ADJUST }

    # --- Asset & Privacy ---
    asset_code: string      # ISO 4217, ISIN, or Contract Address
    privacy_mode: enum { TRANSPARENT, CONFIDENTIAL_AMT, SHIELDED }

    # If transparent:
    amount_scalar: decimal
    # If confidential (ZK):
    amount_commitment: bytes
    proof_ref: string

    # --- Accounts & Buckets ---
    # Can be an address, or a ZK Nullifier/Commitment hash
    src_account: string or bytes or NULL
    src_balance_type: enum { available, reservation, collateral, escrow, channel }

    dst_account: string or bytes or NULL
    dst_balance_type: enum { available, reservation, collateral, escrow, channel }

    # --- Global State Deltas ---
    supply_delta: decimal   # Net change to total supply (for P3)
    lock_delta: decimal     # Net change to encumbered quantity (for P2)

    # --- Linkage & Atomicity ---
    link_id: string         # Groups atomic bundles (e.g. PvP, Flash Loan)
    atomicity: enum { none, PvP, DvP, HTLC, protocol_atomic }

    # --- Evidence / Upstream Messaging (ISO 20022 Alignment) ---
    message_ref: json or NULL  # References to pacs/pain/camt or SWIFT MT
    purpose_code: string       # ISO 20022 <Purp> (e.g., "SALA", "COLL")
```

3.3 Three Irreducible Primitives

Every BSTE is exactly one of three primitives:

P1. OWNERSHIP_TRANSFER

Move spendable units between accounts **without** changing total supply or system-wide encumbrance.

- Constraints: $supply\_delta = 0$, $lock\_delta = 0$.
- Instances: RTGS payment, ACH posting, BTC transfer, Lightning channel rebalance.

P2. ENCUMBRANCE_ADJUST

Change how much balance is **free vs encumbered** (reservation/collateral/escrow) **without** changing ownership or total supply.

- Constraints: $supply\_delta = 0$, $lock\_delta \neq 0$.
- Instances: Card authorisation, Repo collateral pledge, Lightning Harming Logic (HTLC), Optimistic Bridge lock.

P3. SUPPLY_ADJUST

Increase or decrease total recognised supply of a given asset on a given ledger (mint/burn).

- Constraints: $lock\_delta = 0$, $supply\_delta \neq 0$.
- Instances: Central bank reserve creation, Stablecoin mint/burn, Tokenised MMF share issuance, Wrapping/Unwrapping assets.

4 Core Digital Liquidity Profile (C1–C20)

The Profile constitutes a **domain ontology** defined by a finite feature space of 20 fields. Each instrument or infrastructure is defined by a unique tuple in this space.

A core contribution of this profile is the formalisation of **trust**. Rather than labeling systems broadly as “trusted" or “trustless", the C1–C20 dimensions decompose trust into specific vectors: *Solvency Trust* (C1–C3), *Custodial Trust* (C9, C20), *Operational Trust* (C12–C14), and *Governance Trust* (C10, C18). This allows researchers to quantify exactly *what* a user must trust in any given system-e.g., trusting a stablecoin issuer’s solvency (C3) but not their ability to reverse settled transactions (C6).

4.1 Asset Layer (C1–C3)

C1. Asset Nature: What kind of liability is this? *Values:* `cb_reserve`, `cb_cash`, `commercial_deposit`, `e_money`, `stablecoin_fiat`, `token_native`, `security_cash`.

C2. Representation Model: How is the claim stored? *Values:* `native_account`, `native_token`, `wrapped_mirror`, `synthetic`.

C3. Liquidity Backing: What secures the value? *Values:* `unsecured`, `secured_specific_collateral` (Repo/DeFi), `fully_reserved` (Narrow bank/Stablecoin).

4.2 Ledger Layer (C4–C7)

C4. Ledger Tech: Where does the state live? *Values:* cb_rtgs, bank_core, dlt_public, dlt_permissioned, fmi_ledger, channel_state, scheme_internal.

C5. Settlement Mode: How are obligations settled? *Values:* gross (RTGS), multilateral_net_batch (ACH), onchain_gross.

C6. Finality Kind: When is it irreversible? *Values:* deterministic, probabilistic (Nakamoto), deterministic_deferred (Optimistic Rollups).

C7. State Visibility: Who can see the global state? *Values:* public, private_consortium, encrypted_state (ZK).

4.3 Access, Trust & Privacy (C8–C11)

C8. Access Model: Who can hold balances? *Values:* direct_wholesale, retail, permissionless, permissioned.

C9. Custody Model: Who holds the keys? *Values:* self_custody, bank_custody, smart_contract_custody, infrastructure_custody.

C10. Governance Model: Who changes the rules? *Values:* issuer_board, dao_token_vote, immutable_contract, statutory.

C11. Privacy Model: How much information is hidden? *Values:* transparent (Bitcoin), pseudonymous (Ethereum), confidential_amounts (Liquid), fully_shielded (Zcash/Tornado), confidential_routing (Lightning).

4.4 Ordering & Systems Semantics (C12–C16)

C12. Ordering Mechanism: Who decides the sequence of events (MEV)? *Values:* fifo, fee_auction, proposer_privilege (Standard Block Builders), fair_sequencing (Encrypted Mempool).

C13. Liveness Dependency: On whom does progress depend? *Values:* autonomous, sequencer_dependent, operator_dependent.

C14. Safety Fallback: What happens during a partition/failure? *Values:* halt (Consistency favored), fork (Availability favored), escape_hatch (L1 Force Exit).

C15. Cross-Ledger Pattern: How is value moved between ledgers? *Values:* none, lock_mint, burn_mint, htlc, notary_federation, optimistic_bridge.

C16. Messaging Primitive: How are changes driven? *Values:* offledger_messages (ISO 20022), onchain_transactions, hybrid.

4.5 Credit and Systemic Control (C17–C20)

C17. Credit Sources: Where does funding elasticity come from? *Values:* prefunded, intraday_credit, flash_loan, repo, retail_credit.

C18. Upgrade Timelock: How fast can the system rules change? *Values:* none, time_delay, immutable.

C19. Censorship Boundary: At what layer can transactions be blocked? *Values:* network_edge (IP blocking), builder (Tx exclusion), contract (Blacklist logic).

C20. Sanction Capability: What coercive actions can the operator take? *Values:* freeze_account, freeze_asset, none.

5 Alignment with Regulatory Frameworks

5.1 Basel III (LCR / NSFR)

Basel III reasoning hinges on HQLA quality and encumbrance. Our profile supports this via derived metrics:

- **Digital HQLA:** Using `liquidity_backing` and `asset_nature` to classify tiers.
- **Free Liquidity:** $F(t) = Supply(t) - Encumbered(t)$, where Encumbered is derived from P2 events.
- **Funding Structure:** `credit_sources` distinguishes between unsecured interbank funding (run risk) and secured funding (repo/SFT).

5.2 PFMI (CPSS–IOSCO)

- **Principle 8 (Finality):** Mapped directly to `finality_kind` and `safety_fallback`.
- **Principle 5 (Collateral):** P2 events allow on-chain collateral (vaults, channels) to be mapped into PFMI frameworks.
- **Principle 2 (Governance):** Mapped to `governance_model` and `upgrade_timelock`.

5.3 ISO 20022 Alignment

ISO 20022 provides a **message model**, not a ledger state model. BSTE is its **state-side dual**:

- A `pacs.008` settlement leg corresponds to a P1 OWNERSHIP_TRANSFER.
- Securities/corporate action messages correspond to P3 SUPPLY_ADJUST and associated P1 transfers.
- Status reports (`camt.053`) correspond to state read-outs post-BSTE.

This allows DLT instruments to be embedded into ISO 20022 workflows by mapping BSTEs to the appropriate message category.

6 SoK as a State-Machine Type System

Languages like Solidity define implementation; the SoK defines the *intended instrument type*. This allows us to pose the question: *Does the design actually satisfy the obligations implied by its type?*

6.1 The Five-Step Evaluation Pipeline

1. **Classify (C1–C20):** Fill out the profile for the target system. For example, a stablecoin might declare itself `stablecoin_fiat`, `fully_reserved`, `burn_mint` bridge.
2. **Derive Obligations:** A `fully_reserved` stablecoin implies $Supply \leq Reserves$. A `lock_mint` bridge implies $Supply_{wrapped} \leq Locked_{native}$. A `dao_token_vote` governance model implies that logic changes must pass a voting threshold.
3. **Map Primitives:** Map the contract's behavioral surface (functions like `deposit`, `swap`) to sequences of P1/P2/P3 events.
4. **Verify (Model Checking):** Encode the state machine and obligations in a formalism (e.g., TLA+) to check if the primitives satisfy the obligations under all reachable states.
5. **Compare:** Evaluate designs that share the same type but differ in governance or ordering risks (e.g., two stablecoins with different `sanction_capability`).

6.2 Conceptual TLA+ Mapping

The BSTE model is able to be mapped naturally to TLA+ (Temporal Logic of Actions):

- **Variables:** `balance[account]`, `supply`, `reserve`.
- **Actions:**
  - `Transfer(from, to, amt)` $\rightarrow$ P1 (Ownership Transfer).
  - `Mint(amt, to)` $\rightarrow$ P3 (Supply Adjust).
  - `Lock(amt, user)` $\rightarrow$ P2 (Encumbrance Adjust).
- **Invariants:**
  - Conservation: `supply = Sum(balance)`.
  - Backing: `supply <= reserve`.

The model checker then explores all sequences of actions (including adversarial ordering) to find states where invariants like "Backing" are violated.

6.3 Formal Subtyping: Liskov Substitution for Money

We propose a formal definition for instrument equivalence:

> *System S is a subtype of System T ($S <: T$) if S preserves all invariants of T under all failure modes defined in T.*

For example, a Stablecoin is a subtype of 'Commercial Deposit' only if it maintains par redemption under the failure mode of the issuer's insolvency (often requiring bankruptcy remoteness, captured in `custody_model`).

7 System-Level Applications

7.1 Unified Liquidity Monitoring

By instrumenting systems to emit BSTEs, we can build cross-rail observatories. A central bank could monitor $Supply(USD)$ across RTGS, Tokenised Deposits, and Stablecoins in real-time, detecting leaks or unbacked credit expansion.

7.2 Liquidity Calculus and Optimisation

The BSTE stream allows us to define rigorous metrics for digital liquidity optimisation.

7.2.1 Digital LCR/NSFR

For participant i and asset a:

$$\mathrm{LCR}_{i,a}(t) = \frac{\mathrm{HQLA}_{i,a}^{\mathrm{unenc}}(t)}{\mathrm{NetOutflows}_{i,a}(t, t+30d)}$$

where $\mathrm{HQLA}^{\mathrm{unenc}}$ is inferred from C1 (`asset_nature`) and C14 (Encumbrance State derived from P2 events).

7.2.2 Cost of Encumbrance

For liquidity saving mechanisms (LSMs), we can define the cost of encumbrance for participant i as:

$$\text{Cost}_i = \int K_i(t)\,dt$$

where K is the encumbered quantity. The optimisation problem then becomes: Find a sequence of P1/P2/P3 events that satisfies all payment obligations while minimizing $\int K(t)dt$.

7.3 Decomposing Representation Risk and the Synchronisation Operator

A critical ambiguity in digital finance is the distinction between "native" assets (which live on the ledger) and "tokenised" representations of off-chain value (RWAs). The SoK resolves this by formalising the role of the **Synchronisation Operator** - the entity responsible for maintaining consistency between the on-chain state and the external reality.

7.3.1 General Case: Tokenised Representation

Industry terminology often conflates the asset with its record. The SoK forces a structural distinction via `representation_model` (C2):

- **`native_token`:** The ledger state is the definitive reality (e.g., Bitcoin, Ether). No external synchronisation is required; the asset is `autonomous` (C13).
- **`tokenised_representation`:** The ledger is a shadow record of an external state (e.g., Tokenised MMFs, Real Estate, Corporate Bonds).

Classifying an instrument as a `tokenised_representation` logically implies the existence of a Synchronisation Operator. This creates a specific **Safety Invariant** that the system must satisfy, yet cannot enforce via protocol logic alone:

$$\forall t, \quad \text{Supply}_{\text{onchain}}(t) \leq \text{ProvenBalance}_{\text{offchain}}(t)$$

Consequently, any system with `C2: tokenised_representation` must fundamentally accept `operator_dependent` liveness (C13). Even if the blockchain is functional, the asset loses its peg or redeemability if the off-chain operator fails to synchronise the ledger with the external custodian.

7.3.2 Special Case: Synthetic CBDC

This general theory resolves the confusion between **True Wholesale CBDC** and **Synthetic CBDC**. While both aim to provide risk-free settlement assets, the SoK reveals a fundamental ontological difference:

1. **True Wholesale CBDC:** Defined as $\langle$C1:`cb_reserve`, C2:`native_token`, C13:`autonomous`$\rangle$. The asset **is** the central bank liability. The ledger constitutes the definitive legal settlement; no external synchronisation is required.
2. **Synthetic CBDC:** Defined as $\langle$C1:`e_money`, C2:`wrapped_mirror`, C3:`fully_reserved`$\rangle$. The asset is a private claim **backed** by central bank reserves.

By exposing the `tokenised_representation` attribute, the framework highlights that the Synthetic CBDC is structurally identical to a Tokenised RWA. It relies on a private operator to atomically lock reserves in the RTGS when minting tokens on-chain (`cross_ledger_pattern: lock_mint`). This framework implies that "Synthetic CBDC" is a misnomer: it is simply a highly-secure stablecoin that reintroduces the counterparty and operational risks (via the Synchronisation Operator) that a native CBDC is designed to eliminate.

7.4 Power: Messaging vs. Shared Ledgers

The framework clarifies where **power** resides-specifically the power to censor.

7.4.1 SWIFT (Messaging)

- **Profile:** `ledger_tech = none`, `messaging = offledger`, `censorship = network_edge`.
- **Power:** Sanctions occur by preventing messages from flowing. However, the ledger state remains distributed across banks; SWIFT cannot unilaterally mutate a bank's balance sheet.

7.4.2 mBridge / Multi-CBDC (Shared Ledger)

- **Profile:** `ledger_tech = dlt_permissioned`, `messaging = onchain`, `censorship = builder/contract`.
- **Power:** Authorities act as operators of the shared state machine. They can not only block transactions but potentially freeze or redirect assets directly via the consensus mechanism or smart contract logic.

This shift from *messaging coalitions* to *ledger coalitions* represents a fundamental change in the architecture of financial sanctions.

A Appendix A: Complete Case Studies Library

This appendix provides the full 20-dimensional mapping for 20 diverse digital liquidity systems.

A.1: Fedwire (US RTGS)

C1 Asset	`cb_reserve`
C2 Rep	`native_account`
C3 Backing	`fully_reserved`
C4 Ledger	`cb_rtgs`
C5 Settl	`gross`
C6 Finality	`deterministic`
C7 Visible	`private`
C8 Access	`wholesale`
C9 Custody	`bank_custody`
C10 Gov	`statutory`
C11 Privacy	`transparent (op)`
C12 Order	`fifo`
C13 Live	`operator_dep`
C14 Safe	`halt`
C15 X-Ledger	`none`
C16 Msg	`offledger`
C17 Credit	`intraday`
C18 Upgr	`time_delay`
C19 Censor	`network_edge`
C20 Sanct	`freeze_account`

A.2: US ACH (Nacha)

C1 Asset	`comm_deposit`
C2 Rep	`native_account`
C3 Backing	`unsecured`
C4 Ledger	`bank_core`
C5 Settl	`net_batch`
C6 Finality	`deferred`
C7 Visible	`private`
C8 Access	`retail`
C9 Custody	`bank_custody`
C10 Gov	`consortium`
C11 Privacy	`transparent (op)`
C12 Order	`fifo`
C13 Live	`operator_dep`
C14 Safe	`halt`
C15 X-Ledger	`none`
C16 Msg	`offledger`
C17 Credit	`unsecured`
C18 Upgr	`consortium`
C19 Censor	`network_edge`
C20 Sanct	`freeze_account`

A.3: Card Schemes (Visa)

C1 Asset	`comm_deposit`
C2 Rep	`native_account`
C3 Backing	`unsecured`
C4 Ledger	`scheme_internal`
C5 Settl	`net_batch`
C6 Finality	`probabilistic`
C7 Visible	`private`
C8 Access	`retail`
C9 Custody	`bank_custody`
C10 Gov	`issuer_board`
C11 Privacy	`transparent (op)`
C12 Order	`fifo`
C13 Live	`operator_dep`
C14 Safe	`halt`
C15 X-Ledger	`none`
C16 Msg	`offledger`
C17 Credit	`retail_credit`
C18 Upgr	`issuer`
C19 Censor	`network_edge`
C20 Sanct	`freeze_account`

A.4: CLS (FX PvP)

C1 Asset	`security_cash`
C2 Rep	`native_account`
C3 Backing	`fully_reserved`
C4 Ledger	`fmi_ledger`
C5 Settl	`gross`
C6 Finality	`deterministic`
C7 Visible	`private`
C8 Access	`wholesale`
C9 Custody	`bank_custody`
C10 Gov	`consortium`
C11 Privacy	`transparent (op)`
C12 Order	`fifo`
C13 Live	`operator_dep`
C14 Safe	`halt`
C15 X-Ledger	`proto_atomic`
C16 Msg	`offledger`
C17 Credit	`intraday`
C18 Upgr	`consortium`
C19 Censor	`network_edge`
C20 Sanct	`freeze_account`

A.5: CCP Cash (LCH/ICE)

C1 Asset	`security_cash`
C2 Rep	`native_account`
C3 Backing	`fully_reserved`
C4 Ledger	`fmi_ledger`
C5 Settl	`net_batch`
C6 Finality	`deterministic`
C7 Visible	`private`
C8 Access	`wholesale`
C9 Custody	`bank_custody`
C10 Gov	`statutory`
C11 Privacy	`transparent (op)`
C12 Order	`fifo`
C13 Live	`operator_dep`
C14 Safe	`halt`
C15 X-Ledger	`none`
C16 Msg	`offledger`
C17 Credit	`repo`
C18 Upgr	`statutory`
C19 Censor	`network_edge`
C20 Sanct	`freeze_account`

A.6: Bitcoin (L1)

C1 Asset	`token_native`
C2 Rep	`utxo`
C3 Backing	`unsecured`
C4 Ledger	`dlt_public`
C5 Settl	`onchain_gross`
C6 Finality	`probabilistic`
C7 Visible	`public`
C8 Access	`permissionless`
C9 Custody	`self_custody`
C10 Gov	`immutable`
C11 Privacy	`pseudonymous`
C12 Order	`fee_auction`
C13 Live	`autonomous`
C14 Safe	`fork`
C15 X-Ledger	`none`
C16 Msg	`onchain_tx`
C17 Credit	`prefunded`
C18 Upgr	`immutable`
C19 Censor	`builder`
C20 Sanct	`none`

A.7: Lightning Network

C1 Asset: `token_native`
C2 Rep: `hybrid`
C3 Backing: `fully_reserved`
C4 Ledger: `channel_state`
C5 Settl: `gross`
C6 Finality: `local_determ`
C7 Visible: `pvt_channels`
C8 Access: `permissionless`
C9 Custody: `self_custody`
C10 Gov: `immutable`
C11 Privacy: `conf_routing`
C12 Order: `fifo`
C13 Live: `autonomous`
C14 Safe: `halt`
C15 X-Ledger: `htlc`
C16 Msg: `hybrid`
C17 Credit: `prefunded`
C18 Upgr: `immutable`
C19 Censor: `none`
C20 Sanct: `none`

A.8: Ethereum (L1)

C1 Asset: `token_native`
C2 Rep: `account`
C3 Backing: `unsecured`
C4 Ledger: `dlt_public`
C5 Settl: `onchain_gross`
C6 Finality: `probabilistic`
C7 Visible: `public`
C8 Access: `permissionless`
C9 Custody: `self_custody`
C10 Gov: `consortium`
C11 Privacy: `pseudonymous`
C12 Order: `prop_privilege`
C13 Live: `autonomous`
C14 Safe: `fork`
C15 X-Ledger: `none`
C16 Msg: `onchain_tx`
C17 Credit: `prefunded`
C18 Upgr: `immutable`
C19 Censor: `builder`
C20 Sanct: `none`

A.9: Optimistic Rollup

C1 Asset: `wrapped_mirror`
C2 Rep: `native_account`
C3 Backing: `secured`
C4 Ledger: `dlt_public`
C5 Settl: `onchain_gross`
C6 Finality: `deferred`
C7 Visible: `public`
C8 Access: `permissionless`
C9 Custody: `contract`
C10 Gov: `dao_vote`
C11 Privacy: `pseudonymous`
C12 Order: `prop_privilege`
C13 Live: `sequencer_dep`
C14 Safe: `escape_hatch`
C15 X-Ledger: `opt_bridge`
C16 Msg: `onchain_tx`
C17 Credit: `prefunded`
C18 Upgr: `time_delay`
C19 Censor: `builder`
C20 Sanct: `none`

A.10: USDT (Tether)

C1 Asset: `stable_fiat`
C2 Rep: `native_token`
C3 Backing: `secured`
C4 Ledger: `dlt_public`
C5 Settl: `onchain_gross`
C6 Finality: `probabilistic`
C7 Visible: `public`
C8 Access: `permissionless`
C9 Custody: `self_custody`
C10 Gov: `issuer_board`
C11 Privacy: `pseudonymous`
C12 Order: `fee_auction`
C13 Live: `sequencer_dep`
C14 Safe: `halt`
C15 X-Ledger: `lock_mint`
C16 Msg: `onchain_tx`
C17 Credit: `prefunded`
C18 Upgr: `none`
C19 Censor: `contract`
C20 Sanct: `freeze_account`

A.11: USDC (Circle)

C1 Asset: `stable_fiat`
C2 Rep: `native_token`
C3 Backing: `fully_res`
C4 Ledger: `dlt_public`
C5 Settl: `onchain_gross`
C6 Finality: `probabilistic`
C7 Visible: `public`
C8 Access: `permissionless`
C9 Custody: `self_custody`
C10 Gov: `issuer_board`
C11 Privacy: `pseudonymous`
C12 Order: `fee_auction`
C13 Live: `sequencer_dep`
C14 Safe: `halt`
C15 X-Ledger: `burn_mint`
C16 Msg: `hybrid`
C17 Credit: `prefunded`
C18 Upgr: `time_delay`
C19 Censor: `contract`
C20 Sanct: `freeze_account`

A.12: Tokenised Deposits

C1 Asset: `comm_deposit`
C2 Rep: `wrapped_mirror`
C3 Backing: `unsecured`
C4 Ledger: `dlt_perm`
C5 Settl: `onchain_gross`
C6 Finality: `probabilistic`
C7 Visible: `private`
C8 Access: `permissioned`
C9 Custody: `bank_custody`
C10 Gov: `issuer_board`
C11 Privacy: `conf_amt`
C12 Order: `prop_privilege`
C13 Live: `operator_dep`
C14 Safe: `halt`
C15 X-Ledger: `burn_mint`
C16 Msg: `hybrid`
C17 Credit: `unsecured`
C18 Upgr: `time_delay`
C19 Censor: `contract`
C20 Sanct: `freeze_account`

A.13: RWA Vault (4626)

C1 Asset: security_cash
C2 Rep: native_token
C3 Backing: secured
C4 Ledger: dlt_public
C5 Settl: onchain_gross
C6 Finality: probabilistic
C7 Visible: public
C8 Access: permissionless
C9 Custody: contract
C10 Gov: dao_vote
C11 Privacy: pseudonymous
C12 Order: fee_auction
C13 Live: sequencer_dep
C14 Safe: halt
C15 X-Ledger: none
C16 Msg: onchain_tx
C17 Credit: flash_loan
C18 Upgr: time_delay
C19 Censor: contract
C20 Sanct: none

A.14: Tokenised MMF

C1 Asset: security_cash
C2 Rep: native_token
C3 Backing: secured
C4 Ledger: dlt_public
C5 Settl: onchain_gross
C6 Finality: probabilistic
C7 Visible: public
C8 Access: kyc_gated
C9 Custody: bank_custody
C10 Gov: issuer_board
C11 Privacy: pseudonymous
C12 Order: fee_auction
C13 Live: sequencer_dep
C14 Safe: halt
C15 X-Ledger: burn_mint
C16 Msg: hybrid
C17 Credit: prefunded
C18 Upgr: time_delay
C19 Censor: contract
C20 Sanct: freeze_account

A.15: Uniswap v3 Pool

C1 Asset: token_native
C2 Rep: native_token
C3 Backing: secured
C4 Ledger: dlt_public
C5 Settl: onchain_gross
C6 Finality: probabilistic
C7 Visible: public
C8 Access: permissionless
C9 Custody: contract
C10 Gov: immutable
C11 Privacy: pseudonymous
C12 Order: fee_auction
C13 Live: autonomous
C14 Safe: halt
C15 X-Ledger: none
C16 Msg: onchain_tx
C17 Credit: prefunded
C18 Upgr: immutable
C19 Censor: builder
C20 Sanct: none

A.16: Aave Lending

C1 Asset: wrapped_mirror
C2 Rep: native_token
C3 Backing: secured
C4 Ledger: dlt_public
C5 Settl: onchain_gross
C6 Finality: probabilistic
C7 Visible: public
C8 Access: permissionless
C9 Custody: contract
C10 Gov: dao_vote
C11 Privacy: pseudonymous
C12 Order: fee_auction
C13 Live: autonomous
C14 Safe: halt
C15 X-Ledger: none
C16 Msg: onchain_tx
C17 Credit: overcollat
C18 Upgr: time_delay
C19 Censor: contract
C20 Sanct: none

A.17: CeFi (Binance)

C1 Asset: comm_deposit
C2 Rep: native_account
C3 Backing: unsecured
C4 Ledger: scheme_int
C5 Settl: gross
C6 Finality: deterministic
C7 Visible: private
C8 Access: retail
C9 Custody: infra_custody
C10 Gov: issuer_board
C11 Privacy: private
C12 Order: fifo
C13 Live: operator_dep
C14 Safe: halt
C15 X-Ledger: none
C16 Msg: offledger
C17 Credit: margin
C18 Upgr: none
C19 Censor: network_edge
C20 Sanct: freeze_account

A.18: AliPay

C1 Asset: e_money
C2 Rep: native_account
C3 Backing: fully_res
C4 Ledger: scheme_int
C5 Settl: gross
C6 Finality: deterministic
C7 Visible: private
C8 Access: retail
C9 Custody: bank_custody
C10 Gov: issuer_board
C11 Privacy: private
C12 Order: fifo
C13 Live: operator_dep
C14 Safe: halt
C15 X-Ledger: none
C16 Msg: offledger
C17 Credit: retail_credit
C18 Upgr: none
C19 Censor: network_edge
C20 Sanct: freeze_account

A.19: e-CNY (CBDC)

C1 Asset: `cb_cash`
C2 Rep: `native_account`
C3 Backing: `fully_res`
C4 Ledger: `dlt_perm`
C5 Settl: `gross`
C6 Finality: `deterministic`
C7 Visible: `private`
C8 Access: `retail`
C9 Custody: `bank_custody`
C10 Gov: `statutory`
C11 Privacy: `managed`
C12 Order: `fifo`
C13 Live: `operator_dep`
C14 Safe: `halt`
C15 X-Ledger: `none`
C16 Msg: `offledger`
C17 Credit: `none`
C18 Upgr: `statutory`
C19 Censor: `network_edge`
C20 Sanct: `freeze_account`

A.20: Tornado Cash

C1 Asset: `wrapped_mirror`
C2 Rep: `account`
C3 Backing: `secured`
C4 Ledger: `dlt_public`
C5 Settl: `gross`
C6 Finality: `probabilistic`
C7 Visible: `public`
C8 Access: `permissionless`
C9 Custody: `contract`
C10 Gov: `immutable`
C11 Privacy: `shielded`
C12 Order: `fee_auction`
C13 Live: `autonomous`
C14 Safe: `fork`
C15 X-Ledger: `lock_mint`
C16 Msg: `onchain_tx`
C17 Credit: `prefunded`
C18 Upgr: `immutable`
C19 Censor: `builder`
C20 Sanct: `none`

B Appendix B: Semantic Formalisation Strategy

While the Core Digital Liquidity Profile (C1–C20) provides a conceptual ontology for distinguishing instruments and infrastructures, formalising this structure into the Web Ontology Language (OWL 2) offers significant advantages for automated verification and interoperability. This appendix outlines the motivation and mapping strategy for translating the semi-formal SoK into a machine-readable artifact.

B.1 Motivation for an OWL Artifact

Defining the SoK as a formal OWL ontology enables three capabilities that are not possible with a textual taxonomy alone:

1. **Automated Consistency Checking:** By defining constraints using Description Logic (DL), off-the-shelf reasoners (e.g., HermiT, Pellet) can automatically detect logical contradictions. For example, a system that declares itself as `ledger_tech: dlt_public` but claims `finality_kind: deterministic` would flag a logical inconsistency if the ontology defines public chains as inherently probabilistic.
2. **Subsumption and Classification:** We can formally define instrument types (e.g., "Valid Stablecoin") as classes with specific restrictions. The reasoner can then automatically classify specific instances (e.g., *USDC*) based on their properties, validating whether they truly meet the criteria of their claimed type.
3. **Interoperability and Querying:** An OWL representation allows the risk profile of financial infrastructures to be queried using SPARQL, integrating this data into broader Knowledge Graphs used by regulators or risk-management dashboards.

B.2 Mapping Strategy

The transition from the tabular SoK profile to OWL primitives follows a direct structural mapping. The domain of discourse is divided into *Classes* (the entities), *Object Properties* (the dimensions), and *Named Individuals* (the values and case studies).

B.2.1 Structural Mapping Table

The proposed mapping between the SoK components and OWL 2 primitives is defined as follows:

SoK Component	**OWL 2 Primitive**	**Description**
System / Instrument	`owl:Class`	The top-level concepts (e.g., *DigitalInstrument*, *Infrastructure*).
Dimensions (C1–C20)	`owl:ObjectProperty`	The relationships connecting an instrument to its attributes (e.g., *hasAssetNature*, *hasFinalityKind*).
Dimension Values	`owl:NamedIndividual`	The specific enumerated values allowed for each dimension (e.g., *fully_reserved*, *dlt_public*).
Case Studies	`owl:NamedIndividual`	Concrete instances of systems (e.g., *Fedwire*, *TornadoCash*) instantiating the classes.
Type Obligations	`owl:Restriction`	Logical constraints that define necessary conditions for a class (e.g., "A Stablecoin must be fully reserved").

Table 1: Mapping Strategy from SoK Profile to OWL 2 DL

B.3 Encoding Invariants as Restrictions

The “Type System” aspect of this SoK is realised in OWL through `owl:equivalentClass` and property restrictions. This allows us to formalise the obligations derived in Section 6.

For example, to enforce the invariant that a “True Stablecoin” must be fully reserved, we define the class not merely as a label, but as a logical restriction:

```
:ValidStablecoin rdf:type owl:Class ;
    owl:equivalentClass [
        rdf:type owl:Class ;
        owl:intersectionOf (
            :DigitalInstrument
            [ rdf:type owl:Restriction ;
              owl:onProperty :hasAssetNature ;
              owl:hasValue :stablecoin_fiat ]
            [ rdf:type owl:Restriction ;
              owl:onProperty :hasLiquidityBacking ;
              owl:hasValue :fully_reserved ]
        )
    ] .
```

Listing 1: Conceptual Turtle Syntax for Type Definition

Under this formalisation, if a case study instance (e.g., *AlgorithmicCoin*) is instantiated with `hasAssetNature: stablecoin_fiat` but `hasLiquidityBacking: unsecured`, the reasoner will correctly infer that it is *not* a member of the `:ValidStablecoin` class, regardless of its marketing labels.

This formalisation strategy ensures that the “Core Digital Liquidity Profile” acts as a rigorous semantic standard rather than a subjective classification scheme.