CATHEXIS: A Trust-Handle Layer for EQBSL Networks

A Yellowpaper for the Categoriser + LLM Labeling System

Steake Co-pilot (LLM)

Version 0.2 – Draft

0. Status

Version: 0.2 (draft)
Scope: Conceptual + technical design, suitable for first implementation
Intended stack: EQBSL / EBSL trust layer + hypergraph / graph backend + NN categoriser + LLM handle generator

1 Abstract

EQBSL/EBSL-style trust engines produce dense, high-dimensional tensor states over dynamic social and economic networks. Humans do not. Humans traffic in compact, qualitative handles such as "reliable OTC seller", "slow-rug NFT team", or "ecosystem steward".

This yellowpaper specifies **CATHEXIS**, a two-stage abstraction layer:

1. A **Categoriser Network** that ingests
   - agent-level trust signals from an EQBSL/EBSL engine,
   - structural features from the underlying (hyper)graph,
   - behavioural features extracted from protocol-specific activity,

   and maps them to latent, reusable *trust categories*.
2. A **Labeling LLM** that
   - consumes summary statistics and canonical patterns for each category,
   - emits human-readable, irreducible conceptual labels ("handles"),
   - maintains label stability over time as the network evolves.

2 Background: EBSL and EQBSL

CATHEXIS is not a standalone reputation system. It assumes that a trust calculus already exists underneath. In this stack, that calculus is provided by **Evidence-Based Subjective Logic (EBSL)** and its extension, **EQBSL**.

2.1 Subjective Logic in one paragraph

Classical Subjective Logic represents an opinion of agent A about proposition X as

$$\omega_X^A = (b, d, u, a) \tag{1}$$

where

- b = belief,
- d = disbelief,
- u = uncertainty,
- a = base rate (prior probability of X if evidence is uninformative),

with the constraint

$$b + d + u = 1. \tag{2}$$

Subjective Logic defines operators for
- consensus (combining independent opinions),
- recommendation / trust transitivity (A trusts B, B trusts C implies A has an opinion on C),
- discounting (weighting opinions by trust in the source),
- fusion (combining multiple sources into a single opinion).

2.2 Evidence-Based Subjective Logic (EBSL)

EBSL reframes opinions explicitly in terms of evidence counts. For a binary proposition we interpret
- r = amount of positive evidence,
- s = amount of negative evidence,

and a fixed normalisation constant $K > 0$.

We map evidence to a Subjective Logic opinion via

$$b = \frac{r}{r + s + K}, \qquad d = \frac{s}{r + s + K}, \qquad u = \frac{K}{r + s + K}. \tag{3}$$

Key properties:
- Evidence is additive: combining independent observations just adds evidence vectors (r, s).
- The uncertainty term u naturally decays as the total evidence $r + s$ grows.
- Operators like consensus and discounting can be implemented in evidence space, then mapped back to opinions.

On a network:
- Nodes V are agents.
- Edges E carry evidence about interactions (successful trade, failed settlement, timely delivery, malicious proposal, etc.).
- The engine iteratively propagates, aggregates, and discounts evidence along the graph to produce trust opinions per pair or per node.

2.3 EQBSL: an extended, operator view

For CATHEXIS, we need something more structured than "just a bag of opinions". We want
- vectorised trust states we can feed directly into neural networks,
- consistent handling of time, hyperedges, and context,
- a way to define operators over trust that behave like linear (or quasi-linear) maps over evidence tensors.

Definition (EQBSL). EQBSL is an extension of EBSL that lifts evidence-based opinions into a structured, vectorised operator form over dynamic graphs and hypergraphs, with explicit temporal and contextual semantics.

Concretely, EQBSL introduces:

Evidence tensors. Instead of scalar (r, s) for each relationship, we maintain a richer evidence vector

$$\mathbf{e}_{ij}(t) \in \mathbb{R}^m \tag{4}$$

for each ordered pair of agents i, j, where components might include counts of successful vs failed trades, on-time vs late delivery, governance alignment votes, dispute outcomes, attestation patterns, and time-weighted recency scores.

A mapping

$$\Psi : \mathbf{e}_{ij}(t) \mapsto \omega_{ij}(t) = (b_{ij}(t), d_{ij}(t), u_{ij}(t), a_{ij}) \tag{5}$$

lifts these vectors into Subjective Logic / EBSL opinions.

Operator view of trust propagation. Instead of ad-hoc "iterate until convergence" rules, EQBSL regards propagation as an operator

$$\mathcal{F} : \mathcal{E}_t \to \mathcal{E}_{t+1} \tag{6}$$

where $\mathcal{E}_t$ is the set of all evidence tensors at time t. In practice

$$\mathcal{E}_{t+1} = \mathcal{F}(\mathcal{E}_t, \text{new\_events}_t), \tag{7}$$

where $\mathcal{F}$ encodes EBSL combination rules, temporal decay, hyperedge aggregation, and context-dependent weighting.

Hypergraph-aware trust. For multi-party interactions (DAOs, multi-sig, group swaps), EQBSL represents evidence per hyperedge

$$\mathbf{e}_h(t), \qquad h \subseteq V,\ |h| \geq 2, \tag{8}$$

and defines how that evidence decomposes into pairwise or groupwise trust statements.

Node-level trust embeddings. For each node i, EQBSL builds an embedding

$$\mathbf{u}_i(t) = \Gamma(i, \mathcal{E}_t, \mathcal{G}_t) \in \mathbb{R}^{d_u} \tag{9}$$

where Γ aggregates how the network currently "feels" about i across opinions, graph position, and temporal patterns.

These embeddings $\mathbf{u}_i(t)$ are the primary EQBSL output we care about for CATHEXIS.

2.4 Assumptions for CATHEXIS

We assume EQBSL provides:

- per-agent trust embeddings $\mathbf{u}_i(t)$,
- optional pairwise opinions $\omega_{ij}(t)$,
- access to the graph / hypergraph $\mathcal{G}_t$,
- access to evidence tensors,
- a well-defined update operator $\mathcal{F}$.

CATHEXIS sits on top of EQBSL and turns these states into human-facing categories and labels.

3 Formal Model

Let

- V be the set of agents,
- $\mathcal{G}_t = (V, E_t)$ be the time-indexed interaction graph or hypergraph,
- $\mathbf{u}_i(t) \in \mathbb{R}^{d_u}$ be the EQBSL trust embedding for agent i.

For each agent i, define a feature state

$$\mathbf{x}_i(t) \in \mathbb{R}^d \tag{10}$$

assembled from:

a) trust features (embedding $\mathbf{u}_i(t)$, global reputation, uncertainty, etc.),
b) graph features (degree, centrality, clustering, hyperedge signatures),
c) behavioural features (platform-specific metrics, temporal statistics).

We write

$$\mathbf{x}_i(t) = \Phi\big(i, \mathcal{G}_{[0,t]}, \{\mathbf{u}_j(\cdot)\}_{j \in V}\big). \tag{11}$$

We posit a discrete set of latent categories

$$\mathcal{C} = \{1, 2, \ldots, K\} \tag{12}$$

and allow mixture assignments

$$\mathbf{p}_i(t) \in \Delta^{K-1}, \tag{13}$$

the probability simplex over K categories.

A categoriser network

$$f_\theta : \mathbb{R}^d \to \Delta^{K-1} \tag{14}$$

maps feature vectors to category distributions, with hard category

$$c_i(t) = \arg\max_k f_\theta(\mathbf{x}_i(t))_k. \tag{15}$$

For each k,

$$S_k(t) = \{i \in V \mid c_i(t) = k\}, \tag{16}$$

$$\boldsymbol{\mu}_k(t) = \frac{1}{|S_k(t)|} \sum_{i \in S_k(t)} \mathbf{x}_i(t). \tag{17}$$

Optionally,

$$\Sigma_k(t) = \mathrm{Cov}\{\mathbf{x}_i(t) \mid i \in S_k(t)\}. \tag{18}$$

4 Network Architecture

Each platform type p has its own extractor Φ_p. We form

$$\mathbf{x}_i(t) = \mathrm{Concat}_p\ \Phi_p(i, \mathrm{events}_p, \mathbf{u}_i(t), \mathcal{G}_{[0,t]}). \tag{19}$$

A simple MLP baseline is

$$f_\theta(\mathbf{x}) = \mathrm{softmax}\big(W_2\, \sigma(W_1 \mathbf{x} + b_1) + b_2\big). \tag{20}$$

A graph / hypergraph neural network variant uses message passing:

$$\mathbf{h}_i^{(0)} = \mathbf{x}_i(t), \tag{21}$$

$$\mathbf{h}_i^{(l+1)} = \sigma\Big(W^{(l)} \cdot \mathrm{AGG}\big(\{\mathbf{h}_j^{(l)} \mid j \in \mathcal{N}(i)\} \cup \{\mathbf{h}_i^{(l)}\}\big)\Big), \tag{22}$$

$$f_\theta(\mathbf{x}_i(t)) = \mathrm{softmax}(W^{(L)}\mathbf{h}_i^{(L)} + b^{(L)}). \tag{23}$$

Training combines supervised, contrastive, and entropy-regularised terms:

$$\mathcal{L} = \lambda_{\text{sup}}\mathcal{L}_{\text{sup}} + \lambda_{\text{contrast}}\mathcal{L}_{\text{contrast}} + \lambda_{\text{entropy}}\mathcal{L}_{\text{entropy}}. \tag{24}$$

5 Labeling LLM

For each category k, we build a summary with: top features, signed deviations from global means, exemplar statistics, and platform provenance. The LLM returns:

- a short handle (e.g. "high-risk flaky OTC counterparty"),
- a one-sentence gloss,
- optional risk / usage guidance.

We track drift in $\boldsymbol{\mu}_k(t)$ and membership to decide when to split, merge, or re-label categories.

6 Pipeline Sketch

Offline batch:

```
def run_cathexis_batch(snapshot_time):
    G_t, U_t = load_graph_and_trust(snapshot_time)

    X = {}
    for i in G_t.nodes():
        X[i] = compute_features(i, G_t, U_t)

    P, C = {}, {}
    for i, x in X.items():
        p_i = f_theta(x)
        P[i] = p_i
        C[i] = argmax(p_i)

    summaries = build_category_summaries(C, X, G_t, U_t)

    for k, summary in summaries.items():
        if needs_new_label(k, summary):
            label, gloss, guidance = call_labeling_llm(summary)
            save_category_label(k, label, gloss, guidance, snapshot_time)

    persist_assignments(C, P, snapshot_time)
    persist_category_metadata(summaries)
```

Online query:

```
def query_agent_handle(agent_id, now):
    G_now, U_now = get_latest_state()
    x = compute_features(agent_id, G_now, U_now)
    p = f_theta(x)
    k = argmax(p)

    label_info = load_label_for_category(k)
    return {
        "category_id": k,
        "probabilities": p,
        "label": label_info["handle"],
        "description": label_info["gloss"],
        "guidance": label_info["guidance"]
    }
```

7 Summary

CATHEXIS provides a bridge from EQBSL trust embeddings to human-usable trust handles. It is modular, extensible, evaluable, and aimed squarely at the messy trust-adjacent parts of decentralised systems where human judgement and cryptographic guarantees collide.