

Causal Topology Detection: A Framework of Algorithms for Enterprise Causal Intelligence

Author: V **Affiliation:** VectorPeak **Date:** April 19, 2026 **Series:** Meta Generative Principle — Applied Frameworks, No. 1 **License:** CC BY 4.0

Related Publications:

- Meta Generative Principle: <https://doi.org/10.5281/zenodo.19549127>
- GHA and SO Algorithms: <https://doi.org/10.5281/zenodo.19582200>
- The Causal Imperative: <https://doi.org/10.5281/zenodo.19615588>
- The Inevitable Mind: <https://doi.org/10.5281/zenodo.19616056>
- The Generative Ontology: <https://doi.org/10.5281/zenodo.19616056>

Abstract

Enterprise organisations are causal systems of extraordinary complexity. The artificial intelligence frameworks deployed to manage them have universally failed to model this causal complexity — operating instead on statistical correlations in observed data, without access to the generative structure that produces those observations. This paper introduces **Causal Topology Detection** — a systematic framework of nine algorithms for identifying the fundamental patterns of causal dependence that govern enterprise systems. Grounded in the Meta-Generative Principle and implemented through VectorPeak's Generative Hierarchy Algorithm (GHA) and Sustainability Optimizer (SO), the framework identifies eight fundamental causal topologies — Linear, Divergent, Convergent, Cyclical, Hierarchical, Mesh, Latent, and Interventional — and a meta-algorithm, the Hybrid Topology Classifier, that identifies the unique causal topology signature of an enterprise. Each topology is defined precisely, its detection algorithm specified formally, and its enterprise applications developed in detail. Together, the nine algorithms constitute the first systematic framework for causal topology detection in enterprise systems — enabling AI that understands the generative architecture of organisations rather than merely approximating their statistical surface.

Keywords: Causal Topology, Causal AI, Enterprise AI, Meta-Generative Principle, Generative Hierarchy Algorithm, Sustainability Optimizer, Causal Detection, Structural Causal Models, Causal Graph, Hybrid Topology, Enterprise Intelligence, Financial Services AI, Causal Inference, VectorPeak

1. Introduction

1.1 The Causal Architecture Problem

Every enterprise organisation is, at its deepest level, a causal system. Its outcomes — revenues, losses, failures, successes, compliance breaches, operational disruptions — are not statistical events. They are the products of causal chains, feedback loops, convergent factor assemblies, and hierarchical generative processes that constitute the organisation's actual operating architecture.

The artificial intelligence systems deployed to manage enterprise organisations have never modelled this causal architecture. They operate on statistical correlations — learned from historical data, calibrated to historical distributions, and evaluated against historical benchmarks. They learn what tends to follow what. They do not learn what causes what.

This distinction is not merely academic. It is the root cause of the most persistent and costly failures of enterprise AI:

- Fraud detection systems that are perpetually one scheme behind because they detect historical patterns rather than causal mechanisms
- Risk models that fail precisely when they are most needed because they have learned the distribution of stable conditions rather than the mechanisms of stress
- Operational AI that loops on failures it cannot diagnose because diagnosis requires causal attribution and statistical systems have none
- Regulatory compliance tools that cannot explain their decisions because explanation requires causal reasoning and their architectures contain no causal representation

The solution is not better statistical AI. It is a fundamentally different kind of AI — one that models the causal architecture of the enterprise explicitly, discovers the causal topologies that govern its dynamics, and reasons at the level of generative mechanism rather than statistical surface.

This paper introduces the framework that makes this possible.

1.2 Causal Topologies — Definition and Motivation

A **causal topology** is the pattern of generative dependence between entities in a system — the structure of what causes what, with what directionality, at what level of abstraction, with what feedback dynamics, and with what hidden drivers.

The concept of causal topology is motivated by analogy with network topology — the pattern of connections between nodes in a communication system — but represents a fundamentally more important structural property. Network topology describes connectivity. Causal topology describes generativity — the structure of what produces what.

The Meta-Generative Principle (MGP), developed in prior work, establishes that causal structure is ontologically prior to statistical correlation — that the generative structure of reality is more fundamental than any statistical description of its outputs. Causal topology is the application of this ontological priority to the analysis of enterprise systems: before asking what correlations exist in enterprise data, we must ask what causal structure generates that data.

VectorPeak's research has identified eight fundamental causal topologies that, in various combinations, characterise every domain of every enterprise organisation. These eight topologies — Linear, Divergent, Convergent, Cyclical, Hierarchical, Mesh, Latent, and Interventional — are not exhaustive of all possible causal structures but constitute the fundamental patterns from which the causal architectures of real enterprise systems are composed.

1.3 The Detection Problem

An enterprise organisation does not arrive with its causal topology labelled. The topology must be discovered from data, behaviour, and structure — automatically, continuously, and at the scale of enterprise operations. This is the **causal topology detection problem**:

Given a stream of enterprise events, a set of observed variables, and partial domain knowledge, **detect** which causal topology or combination of topologies governs the generative structure of each enterprise domain.

This is not a classification problem in the machine learning sense. It is a **causal structure identification problem** — requiring methods from causal inference, structural equation modelling, and do-calculus reasoning that are qualitatively different from statistical pattern recognition.

1.4 Contributions of This Paper

This paper makes the following contributions:

1. We introduce the concept of **causal topology** as a fundamental analytical category for enterprise AI systems
2. We define eight fundamental causal topologies and provide precise formal characterisations of each
3. We specify nine causal topology detection algorithms — one per topology plus a meta-algorithm for hybrid topology identification
4. We develop the enterprise applications of each algorithm in detail
5. We introduce the concept of the **enterprise causal topology signature** — the unique fingerprint of an organisation's causal architecture
6. We situate the framework within the GHA and SO implementation architecture of the Meta-Generative Principle

1.5 Structure of the Paper

Section 2 presents the theoretical foundations. Section 3 through Section 10 develop each of the eight topology detection algorithms in detail. Section 11 presents the Hybrid Topology Classifier meta-algorithm. Section 12 introduces the enterprise causal topology signature. Section 13 discusses implementation architecture. Section 14 addresses limitations and future directions. Section 15 concludes.

2. Theoretical Foundations

2.1 The Meta-Generative Principle and Causal Priority

The Meta-Generative Principle proposes that reality is a self-generating relational process optimising toward sustainability, in which causal structure is ontologically prior to statistical correlation. This ontological priority has a direct and precise implication for enterprise AI: the causal structure of an enterprise system is more fundamental than any statistical description of its behaviour, and AI systems that model causal structure will systematically outperform those that model statistical correlation — not marginally but categorically.

The Generative Ontology, a further development of the MGP, establishes that reality is a dynamic stream of symbols continuously generated by an algorithm — the meta field — whose generative process is prior to spacetime, matter, energy, and mathematics as we know them. In this framework, causal topology is the structural property of the generative algorithm itself — not a relationship between events in the generated output but a property of the process that generates those events.

For enterprise systems, this means that the causal topology of an organisation is a property of its generative architecture — the structural mechanisms that produce its outcomes — rather than a

pattern in its historical data. Detecting causal topology is discovering something real about the organisation's generative structure, not merely identifying patterns in its statistical surface.

2.2 Structural Causal Models

The formal foundation for causal topology detection is the Structural Causal Model (SCM) framework developed by Pearl (2000). An SCM is a 4-tuple $M = \langle U, V, F, P(U) \rangle$ where:

- **U** is a set of exogenous variables with joint distribution $P(U)$
- **V** = $\{V_1, \dots, V_n\}$ is a set of endogenous variables
- **F** = $\{f_1, \dots, f_n\}$ is a set of structural equations: $V_i = f_i(\text{PA}(V_i), U_i)$
- **PA(V_i)** are the causal parents of V_i in the causal graph

Each SCM induces a causal DAG G whose structure constitutes the causal topology of the system. Causal topology detection is the problem of identifying G — or the class of causal topologies that G instantiates — from observational data, interventional data, and domain knowledge.

2.3 The Causal Hierarchy and Detection

Pearl's causal hierarchy distinguishes three levels of causal reasoning:

- **Level 1 — Observational:** $P(Y | X)$ — statistical association
- **Level 2 — Interventional:** $P(Y | \text{do}(X))$ — causal effect of action
- **Level 3 — Counterfactual:** $P(Y_x | e)$ — what would have happened

Causal topology detection operates primarily at Level 2 — using interventional reasoning to identify causal structure beyond what observational data alone can establish. The algorithms presented in this paper use a combination of observational data, natural experiments, proxy variables, and where available, controlled interventions to identify causal topology with appropriate confidence.

2.4 The GHA and SO as Detection Infrastructure

VectorPeak's Generative Hierarchy Algorithm provides the architectural substrate for causal topology detection. The GHA builds and continuously updates a causal world model of the enterprise — a directed graph of causal relationships at multiple levels of abstraction. When the GHA detects causal representational insufficiency — the condition where its current architecture cannot represent a causal mechanism implied by the data — it autonomously extends its architecture to accommodate the new mechanism.

This self-extension property is essential for causal topology detection in dynamic enterprise environments: as enterprise causal topologies evolve — new feedback loops forming, new latent causes emerging, new intervention points being established — the GHA detects the topological change and extends its representation accordingly.

The Sustainability Optimizer directs the GHA's topology detection and modelling effort by generative efficiency — the richness of causal understanding produced relative to the computational cost of producing it. This ensures that the detection algorithms are deployed with appropriate computational intensity in proportion to the causal significance of the topology being detected.

3. Algorithm 1: Linear Topology Detection — The Causal Chain Tracer (CCT)

3.1 Definition

Linear Causal Topology is the simplest fundamental causal structure: a strictly directed sequence of causal relationships in which each entity has at most one direct upstream cause and at most one direct downstream effect, forming an unbranching causal chain.

Formal definition: A causal graph G exhibits linear topology over a subset $S \subseteq V$ of variables if and only if for every $V_i \in S$: $|PA_G(V_i) \cap S| \leq 1$ and $|CH_G(V_i) \cap S| \leq 1$, where PA_G and CH_G denote causal parents and children in G respectively.

Structural properties:

- Maximal causal traceability — attribution is unambiguous
- Maximal fragility — single point of failure per link
- No causal redundancy — information flows through exactly one path
- Deterministic causal propagation — each cause has exactly one effect

3.2 The Causal Chain Tracer Algorithm

Input: Time series data $X = \{X_1, \dots, X_n\}$, domain knowledge K **Output:** Ordered set of verified causal chains $C = \{C_1, C_2, \dots, C_m\}$

Step 1 — Temporal precedence mapping

For every ordered pair (X_i, X_j) , estimate the temporal precedence relationship using Granger causality testing:

$$F_{X_i \rightarrow X_j} = \frac{RSS(X_j | X_j^{\text{lag}}) - RSS(X_j | X_j^{\text{lag}}, X_i^{\text{lag}})}{RSS(X_j | X_j^{\text{lag}}, X_i^{\text{lag}})}$$

where RSS denotes residual sum of squares and superscript lag denotes lagged values. Pairs where X_i Granger-causes X_j but X_j does not Granger-cause X_i are candidate linear pairs.

Note: Granger causality is used as a temporal precedence filter only — not as a causal claim. Statistical precedence is a necessary but not sufficient condition for causal precedence.

Step 2 — Intervention consistency testing

For each candidate linear pair $(X_i \rightarrow X_j)$, verify causal direction through natural experiment identification — instances in the historical data where X_i was exogenously shifted by an external factor while X_j was not directly affected. Estimate:

$$\hat{\beta}_{i \rightarrow j} = \frac{\Delta \mathbb{E}[X_j]}{\Delta \mathbb{E}[X_i | \text{do}(X_i)]}$$

A consistent, non-zero estimate confirms the causal direction. A null estimate in the reverse direction ($\text{do}(X_j) \rightarrow X_i$) confirms asymmetry.

Step 3 — Mediation exclusion

For each confirmed pair $(X_i \rightarrow X_j)$, test whether the relationship is direct or mediated by an intermediate variable X_k . Using the mediation analysis framework:

$$P(X_j | \text{do}(X_i)) = \sum_{X_k} P(X_j | \text{do}(X_i), X_k = x_k) P(X_k = x_k | \text{do}(X_i))$$

If the direct effect is zero and the total effect is carried entirely through X_k , the true linear chain is $X_i \rightarrow X_k \rightarrow X_j$. Mediation testing extends the chain and ensures correct topology identification.

Step 4 — Chain assembly and validation

Connect verified linear pairs into maximal chains. A chain $c = (V_1 \rightarrow V_2 \rightarrow \dots \rightarrow V_n)$ is valid if:

- Each consecutive pair (V_k, V_{k+1}) is a confirmed causal pair
- No V_k has additional causal parents within S beyond V_{k-1}
- No V_k has additional causal children within S beyond V_{k+1}

Step 5 — Chain criticality scoring

Score each detected chain by:
$$\text{Criticality}(c) = \text{Length}(c) \times \text{AverageStrength}(c) \times \text{BusinessImpact}(c_{\text{terminal}})$$

where BusinessImpact is estimated from domain knowledge K .

3.3 Enterprise Applications

Payment processing chains. The CCT identifies the linear causal chain from payment instruction through clearing, settlement, and reconciliation — enabling precise attribution of settlement failures to their causal origin in the chain.

Trade booking sequences. Linear causal chains from trade execution through booking, confirmation, and position update are identified and monitored — enabling automated detection of broken booking chains before they produce position errors.

Regulatory reporting pipelines. The causal chain from source data through transformation, aggregation, and regulatory submission is identified — enabling precise diagnosis of reporting errors and systematic validation of data lineage.

Operational process monitoring. Any enterprise process with a defined sequential structure — loan origination, claims processing, customer onboarding — exhibits linear causal topology that the CCT can monitor for causal disruptions.

3.4 Output Specification

The CCT produces:

- An ordered catalogue of detected causal chains, ranked by criticality score
- For each chain: the sequence of causal variables, the estimated causal strength of each link, the temporal lag of each link, and the confidence interval of each causal estimate
- A real-time chain integrity monitor that alerts when causal flow in any chain is disrupted

4. Algorithm 2: Divergent Topology Detection — The Root Cause Crystalliser (RCC)

4.1 Definition

Divergent Causal Topology is a causal structure in which a single upstream causal node generates multiple independent downstream effects — a common cause structure in which one root generates a cascade of consequences.

Formal definition: A causal graph G exhibits divergent topology with root R over a subset $S \subseteq V$ if and only if $R \in \text{PA}_G(V_i)$ for all $V_i \in S \setminus \{R\}$, and the elements of $S \setminus \{R\}$ are d-separated from each other given R in G .

Structural properties:

- One cause generates multiple effects
- Effects are causally independent conditional on the root
- Root intervention eliminates all downstream effects simultaneously
- Divergence depth determines the temporal spread of downstream impact

4.2 The Root Cause Crystalliser Algorithm

Input: Anomaly set $A = \{a_1, \dots, a_k\}$ of simultaneous enterprise anomalies, causal graph G_{current}

Output: Ranked list of candidate root causes $R = \{(r_1, s_1), (r_2, s_2), \dots\}$ with confidence scores

Step 1 — Simultaneous anomaly clustering

Define a temporal window τ (typically calibrated to the maximum causal propagation lag in the domain). Group anomalies occurring within τ of each other:

$$\text{Cluster}(A, \tau) = \{a_i, a_j \in A : |t_i - t_j| \leq \tau\}$$

Apply hierarchical clustering to the temporal co-occurrence matrix of anomalies to identify candidate clusters with shared causal origin.

Step 2 — Causal ancestor search

For each anomaly cluster $C = \{a_1, \dots, a_k\}$, search the current causal world model G_{current} for candidate common ancestors:

$$\text{Ancestors}(C) = \bigcap_{a_i \in C} \text{AN}_G(a_i)$$

where $\text{AN}_G(a_i)$ is the set of ancestors of a_i in the causal graph G . Candidate root causes are nodes in $\text{Ancestors}(C)$ that are not themselves ancestors of other nodes in $\text{Ancestors}(C)$ — the most proximate common ancestors.

Step 3 — Conditional independence verification

For each candidate root cause $R \in \text{Ancestors}(C)$, verify the divergent topology signature using the d-separation criterion:

Test whether $\{a_1, \dots, a_k\}$ are mutually conditionally independent given R : $P(a_i, a_j | R) = P(a_i | R) \cdot P(a_j | R) \quad \forall i \neq j \in C$

Candidate roots for which this conditional independence holds are confirmed common causes.

Step 4 — Backdoor adjustment verification

For each confirmed root cause R , estimate the interventional effect using backdoor adjustment:

$$P(a_i | \text{do}(R = r)) = \sum_z P(a_i | R = r, Z = z) P(Z = z)$$

where Z is the set of variables satisfying the backdoor criterion relative to the path $R \rightarrow a_i$. A non-zero, consistent interventional effect across all cluster members confirms the divergent topology.

Step 5 — Root cause ranking

Rank candidate root causes by:
$$\text{Score}(R) = \text{Coverage}(R) \times \text{CausalStrength}(R) \times \text{Confidence}(R)$$

where Coverage is the proportion of cluster anomalies causally explained by R, CausalStrength is the estimated interventional effect magnitude, and Confidence is the statistical confidence of the conditional independence test.

Step 6 — GHA self-extension trigger

If no candidate root cause is found in G_{current} — if the anomaly cluster has a common cause not yet represented in the causal world model — the RCC triggers GHA self-extension: the GHA extends its architecture to represent the new causal source, initialising it from the structural equation implied by the observed downstream effects.

4.3 Enterprise Applications

Model risk management. The RCC identifies model failures as the root cause of simultaneous anomalies across multiple functions consuming the model's output — enabling precise model failure attribution without manual investigation across affected systems.

Counterparty failure impact assessment. When a major counterparty failure produces simultaneous effects across credit exposure, collateral, funding, and client relationships, the RCC identifies the failure as the divergent root and maps the full scope of downstream effects.

Infrastructure failure analysis. Technology infrastructure failures — network outages, database failures, API disruptions — produce divergent downstream effects across multiple business functions. The RCC identifies the infrastructure root and prioritises remediation by downstream impact.

Regulatory breach attribution. When compliance breaches occur simultaneously across multiple regulatory frameworks, the RCC identifies the common upstream cause — a process failure, a control gap, a data quality issue — rather than treating each breach independently.

4.4 Output Specification

The RCC produces:

- A ranked list of candidate root causes with coverage, strength, and confidence scores
- For each root cause: the full set of downstream effects attributed to it, the estimated causal strength of each downstream path, and the recommended remediation priority
- A divergent topology registry — a catalogue of known divergent structures in the enterprise causal graph, updated continuously as new structures are detected

5. Algorithm 3: Convergent Topology Detection — The Multi-Factor Assembly Scanner (MFAS)

5.1 Definition

Convergent Causal Topology is a causal structure in which multiple independent causal factors must be simultaneously present to produce a single outcome — a structure in which no single factor is individually sufficient and the outcome requires the joint presence of multiple necessary conditions.

Formal definition: A causal graph G exhibits convergent topology with outcome O over a set of factors $F = \{F_1, \dots, F_k\}$ if and only if:

- Each $F_i \in \text{PA}_G(O)$ — each factor is a direct causal parent of the outcome
- No proper subset $F' \subset F$ is sufficient to produce O — the outcome requires all factors
- The factors in F are mutually d-separated given the empty set — the factors are causally independent of each other

Structural properties:

- Multiple independent causes converge on a single effect
- No single cause is individually sufficient
- Removing any single factor prevents the outcome
- Simultaneous presence of all factors is jointly sufficient

5.2 The Multi-Factor Assembly Scanner Algorithm

Input: Historical outcome data $O = \{o_1, \dots, o_n\}$, candidate factor set $F = \{F_1, \dots, F_k\}$, current time state S_t

Output: Convergence assembly score $A_t \in [0,1]$ and necessary condition registry NC

Step 1 — Outcome identification and historical analysis

Define the target outcome O (e.g., major loss event, systemic failure, regulatory breach). Identify all historical occurrences $O_{\text{hist}} = \{o : o \in \text{historical data}\}$ and all near-miss occurrences $O_{\text{near}} = \{o : o \text{ approached but did not reach the outcome threshold}\}$.

Step 2 — Necessary condition analysis

For each candidate factor F_i , test whether it is a necessary condition for O :

$$\text{Necessary}(F_i, O) \iff P(O \mid \neg F_i) \approx 0$$

$$\text{Estimate this from historical data: } \hat{P}(O \mid \neg F_i) = \frac{|\{o \in O_{\text{hist}} : F_i \text{ absent at } t_o\}|}{|O_{\text{hist}}|}$$

Factors for which this probability is below a calibrated threshold ϵ are identified as necessary conditions.

Step 3 — Sufficiency testing for factor combinations

For combinations of necessary conditions $F' \subseteq NC$ (where NC is the necessary condition set), test joint sufficiency:

$$\text{Sufficient}(F', O) \iff P(O \mid \bigcap_{F_i \in F'} F_i) \approx 1$$

The minimal sufficient set — the smallest combination of necessary conditions that is jointly sufficient — defines the convergent topology's factor structure.

Step 4 — Causal independence verification

Verify that the identified necessary conditions are causally independent of each other — that no factor causally influences any other factor except through the target outcome. Test pairwise d-separation:

$$\sum_{i \perp G} F_j \quad \forall i \neq j \in NC$$

Factors that are not causally independent require re-examination — their apparent convergence may reflect a shared upstream cause (latent topology) rather than genuine convergence.

Step 5 — Real-time assembly score computation

Deploy the identified convergent topology as a real-time monitoring tool. At each timestep t , compute the assembly score:

$$A_t = \frac{\sum_{i \in NC} w_i \cdot \mathbb{1}[F_i \text{ present at } t]}{\sum_{i \in NC} w_i}$$

where w_i is the estimated causal contribution weight of factor F_i to the outcome O , estimated from the historical interventional effect analysis.

Step 6 — Alert escalation protocol

Define escalation thresholds:

- $A_t > 0.3$ — monitoring alert: convergent assembly beginning
- $A_t > 0.6$ — elevated alert: dangerous assembly in progress
- $A_t > 0.8$ — critical alert: outcome imminent without intervention

Step 7 — Intervention prioritisation

When the assembly score exceeds a threshold, rank available interventions by their causal efficiency:

$$\text{InterventionPriority}(F_i) = \frac{\text{AssemblyScoreReduction}(F_i)}{\text{InterventionCost}(F_i)}$$

The SO optimises this ranking by generative efficiency — selecting the intervention that maximally reduces assembly score relative to its cost.

5.3 Enterprise Applications

Systemic risk prevention. The MFAS monitors the simultaneous assembly of systemic risk factors — excessive leverage, asset price inflation, liquidity mismatch, regulatory gap, correlated exposures — alerting macroprudential authorities when dangerous combinations are forming.

Fraud ring detection. Organised fraud requires the simultaneous presence of multiple necessary conditions — a vulnerable product, a gap in onboarding controls, a transaction monitoring blind spot, and a complicit network. The MFAS identifies the assembly of these conditions before the fraud executes.

Rogue trading prevention. The convergent causal structure of rogue trading incidents — unauthorised risk appetite, risk management gap, supervisory failure, concealment opportunity — is monitored continuously. The MFAS alerts when the assembly score rises toward critical thresholds.

Operational resilience. Major operational failures require the simultaneous failure of primary systems, backup systems, and recovery procedures. The MFAS monitors the assembly of these concurrent failure conditions and alerts when resilience is critically compromised.

5.4 Output Specification

The MFAS produces:

- A necessary condition registry for each defined catastrophic outcome — the set of factors whose simultaneous presence produces the outcome
- A real-time assembly score for each monitored outcome — continuously updated as factor states change
- An alert log with escalation timestamps and intervention recommendations
- A counterfactual analysis for each historical outcome occurrence — identifying which factor, had it been absent, would most efficiently have prevented the outcome

6. Algorithm 4: Cyclical Topology Detection — The Feedback Loop Identifier (FLI)

6.1 Definition

Cyclical Causal Topology is a causal structure in which effects become causes of their own causes — creating feedback dynamics that amplify, stabilise, or oscillate over time.

Formal definition: A causal system exhibits cyclical topology over a subset $S \subseteq V$ if the causal graph G restricted to S contains at least one directed cycle: there exists a sequence $(V_1, V_2, \dots, V_k, V_1)$ such that $V_i \rightarrow V_{i+1} \in E_G$ for all i and $V_k \rightarrow V_1 \in E_G$.

Structural properties:

- Positive feedback (loop gain > 1): amplification dynamics — small perturbations grow
- Negative feedback (loop gain < 1): stabilisation dynamics — perturbations decay
- Marginal feedback (loop gain $= 1$): oscillation dynamics — perturbations persist
- Temporal unrolling required for causal analysis — cycles cannot be represented as DAGs

6.2 The Feedback Loop Identifier Algorithm

Input: Time series data $X = \{X_1(t), \dots, X_n(t)\}$, temporal resolution Δt **Output:** Feedback loop registry $L = \{(\text{cycle}, \text{gain}, \text{phase}, \text{intervention_points})\}$

Step 1 — Bidirectional Granger causality screening

For all pairs (X_i, X_j) , test bidirectional Granger causality. Pairs where both $X_i \rightarrow X_j$ and $X_j \rightarrow X_i$ are significant are candidate feedback pairs:

$$\text{FeedbackCandidate}(X_i, X_j) \iff F_{X_i \rightarrow X_j} > F_{\alpha} \text{ AND } F_{X_j \rightarrow X_i} > F_{\alpha}$$

Step 2 — Cycle enumeration

From the set of feedback candidate pairs, enumerate all directed cycles using depth-first search on the candidate graph. For computational tractability in large enterprise graphs, limit cycle enumeration to cycles of length $\leq L_{\max}$ (typically $L_{\max} = 6$ for enterprise applications).

Step 3 — Loop gain estimation

For each enumerated cycle $C = (V_1 \rightarrow V_2 \rightarrow \dots \rightarrow V_k \rightarrow V_1)$, estimate the loop gain — the amplification factor per cycle traversal:

$$G(C) = \prod_{i=1}^k \beta_{V_i \rightarrow V_{i+1}}$$

where $\beta_{i \rightarrow i+1}$ is the estimated causal coefficient of each link, computed from the structural equation estimation. A gain $G(C) > 1$ indicates positive (amplifying) feedback. $G(C) < 1$ indicates negative (stabilising) feedback.

Step 4 — Temporal unrolling

Convert each detected cycle into a directed acyclic graph (DAG) through temporal unrolling — representing each timestep as a separate layer:

$$V_i(t) \rightarrow V_{i+1}(t + \Delta t_i)$$

where Δt_i is the estimated temporal lag of each causal link. The unrolled DAG enables do-calculus interventional reasoning on the cyclical structure.

Step 5 — Phase detection

For each confirmed feedback loop, identify its current phase using the time series of the loop's key variables:

- **Initiation phase:** Loop gain recently exceeded 1; amplification beginning
- **Growth phase:** Sustained amplification with increasing amplitude
- **Peak phase:** Amplitude approaching historical maxima; reversal risk highest
- **Reversal phase:** Loop gain falling below 1; amplification ending
- **Decay phase:** Sustained deamplification; return toward equilibrium

Phase detection enables intervention timing optimisation — interventions are most effective at specific phases of the feedback cycle.

Step 6 — Optimal intervention point identification

For each feedback loop, identify the variable with maximum intervention leverage — the variable whose modification produces the greatest reduction in loop gain per unit of intervention cost:

$$\text{OptimalIntervention}(C) = \arg\max_{V_i \in C} \frac{\Delta G(C | \text{do}(V_i))}{\text{Cost}(\text{do}(V_i))}$$

6.3 Enterprise Applications

Credit cycle monitoring. The FLI identifies and monitors the credit cycle feedback loop — asset price appreciation → looser credit conditions → increased borrowing → further appreciation — estimating its current gain and phase to provide early warning of cycle reversal.

Liquidity spiral early warning. The FLI monitors the liquidity spiral feedback — falling asset prices → margin calls → forced sales → further price falls — alerting treasury and risk management when the loop gain approaches critical thresholds.

Market momentum and mean reversion. Price momentum (positive feedback) and mean reversion (negative feedback) are identified as cyclical topologies. The FLI's phase detection enables strategy

timing — identifying when momentum is likely to reverse and when mean reversion is likely to complete.

Behavioural feedback in client relationships. Service quality → client satisfaction → retention → revenue → service investment → service quality is a cyclical topology governing client relationship dynamics. The FLI identifies the gain and phase of this loop — enabling investment decisions calibrated to the loop's current dynamics.

6.4 Output Specification

The FLI produces:

- A feedback loop registry: all detected cycles, their gains, current phases, and temporal lags
- Real-time gain monitoring: continuous updates to loop gain estimates as new data arrives
- Phase transition alerts: notifications when a loop transitions between phases
- Intervention recommendations: optimal intervention points and timing for each detected loop

7. Algorithm 5: Hierarchical Topology Detection — The Abstraction Level Mapper (ALM)

7.1 Definition

Hierarchical Causal Topology is a causal structure in which entities and causal relationships exist simultaneously at multiple levels of abstraction, with causation operating both within levels (horizontal causation) and between levels (vertical causation — both upward generation and downward constraint).

Formal definition: A causal system exhibits hierarchical topology over a set of abstraction levels $L = \{L^0, L^1, \dots, L^K\}$ with entity sets $\{V^0, V^1, \dots, V^K\}$ if:

- Intra-level causal relationships exist at each level k : $E_k \subseteq V^k \times V^k$
- Inter-level causal relationships exist between adjacent levels: $E_{\{k,k+1\}} \subseteq V^k \times V^{k+1}$
- At least some inter-level causal effects are irreducible — not fully explained by aggregation of lower-level causal effects

Structural properties:

- Causal relationships are real and irreducible at each level
- Higher levels causally supervene on lower levels (upward causation)
- Higher levels causally constrain lower levels (downward causation)
- Emergent causal properties arise at higher levels that are not present at lower levels

7.2 The Abstraction Level Mapper Algorithm

Input: Enterprise data at multiple granularities $G = \{G^0, G^1, \dots, G^K\}$, domain knowledge K **Output:** Hierarchical causal graph $H = (L, E_L, E_H)$ with intra-level and inter-level causal edges

Step 1 — Granularity stratification

Stratify enterprise variables by their natural level of aggregation, guided by domain knowledge K :

$$\mathcal{L}^k = \{V \in \text{EnterpriseVariables} : \text{AggregationLevel}(V, K) = k\}$$

For financial services: L^0 = transaction level, L^1 = account level, L^2 = portfolio level, L^3 = desk/function level, L^4 = firm level, L^5 = market level.

Step 2 — Intra-level causal discovery

At each abstraction level k , apply standard causal discovery algorithms (PC algorithm, GES, or LiNGAM depending on data characteristics) to identify intra-level causal relationships E_k :

$$E_k = \text{CausalDiscovery}(G^k, \alpha)$$

where α is the significance threshold for conditional independence testing.

Step 3 — Upward causation testing

For each pair of adjacent levels $(k, k+1)$, test whether micro-level causal changes produce macro-level effects beyond simple statistical aggregation:

$$\text{UpwardCausation}(L^k \rightarrow L^{k+1}) \iff \exists V^k \in L^k, V^{k+1} \in L^{k+1} : P(V^{k+1} \mid \text{do}(V^k)) \neq P(V^{k+1} \mid \text{do}(\text{Aggregate}(L^k \setminus V^k)))$$

This tests whether individual lower-level variables have causal effects on higher-level variables that are not captured by aggregate lower-level statistics.

Step 4 — Downward causation testing

Test whether higher-level variables causally constrain lower-level behaviour independently of the lower level's own causal dynamics:

$$\text{DownwardCausation}(L^{k+1} \rightarrow L^k) \iff \exists V^{k+1} \in L^{k+1}, V^k \in L^k : P(V^k \mid \text{do}(V^{k+1}), \text{PA}_{L^k}(V^k)) \neq P(V^k \mid \text{PA}_{L^k}(V^k))$$

This tests whether higher-level interventions produce lower-level effects not fully explained by the lower-level's own causal structure.

Step 5 — Emergence detection

Identify causal properties that emerge at higher abstraction levels and are absent at lower levels. Test for emergence using interventional equivalence:

$$\text{Emergent}(V^{k+1}) \iff \nexists \{V^k_i\} \subseteq L^k : P(O \mid \text{do}(V^{k+1})) = P(O \mid \text{do}(\{V^k_i\}))$$

Emergent causal properties cannot be replicated by any combination of lower-level interventions.

Step 6 — Hierarchical causal graph assembly

Assemble detected intra-level and inter-level causal relationships into the hierarchical causal graph $H = (L, E_L, E_H)$, where E_L are intra-level edges and E_H are inter-level edges.

7.3 Enterprise Applications

Risk aggregation. The ALM identifies genuine cross-level causal effects in risk aggregation — where individual position risks causally generate portfolio-level risks that are not simply the sum of their components. This enables more accurate risk aggregation that preserves cross-level causal dynamics.

Regulatory capital modelling. Capital requirements operate hierarchically — instrument-level capital charges aggregate to portfolio-level requirements which constrain firm-level capital allocation. The ALM identifies the cross-level causal structure of capital dynamics, enabling more accurate capital forecasting.

Enterprise performance management. Individual transaction performance causes function-level performance causes business unit performance causes firm-level performance. The ALM identifies cross-level causal effects and emergent performance dynamics not visible at any single level.

Systemic risk analysis. Systemic risk is an emergent property of the financial system level — causally generated by firm-level interconnections but irreducible to the sum of individual firm risks. The ALM identifies systemic risk as an emergent hierarchical causal property and maps the cross-level mechanisms through which it arises.

7.4 Output Specification

The ALM produces:

- A hierarchical causal map: a multi-level directed graph showing intra-level and inter-level causal relationships
- An emergence registry: emergent causal properties identified at each abstraction level
- Cross-level intervention maps: the causal pathways through which interventions at each level propagate to other levels
- A downward causation monitor: real-time tracking of higher-level causal constraints on lower-level behaviour

8. Algorithm 6: Mesh Topology Detection — The Causal Density Mapper (CDM)

8.1 Definition

Mesh Causal Topology is a causal structure characterised by dense, multi-directional causal interconnection in which multiple entities causally influence multiple other entities through direct and indirect causal paths.

Formal definition: A causal graph G exhibits mesh topology over a subset $S \subseteq V$ if the average causal degree — the mean number of direct causal connections per node — exceeds a domain-calibrated threshold δ :

$$\text{MeshDensity}(S, G) = \frac{\sum_{V_i \in S} (|\text{PA}_G(V_i) \cap S| + |\text{CH}_G(V_i) \cap S|)}{2|S|} > \delta$$

Structural properties:

- High causal redundancy — effects propagate through multiple paths
- Non-linear contagion dynamics — highly connected nodes amplify effects disproportionately
- Global intervention effects — actions at one node propagate throughout the mesh
- Systemic risk as a topological property — not a node property but a mesh property

8.2 The Causal Density Mapper Algorithm

Input: Enterprise variable set V , observational data X , domain knowledge K **Output:** Causal density map $M = (V, E, \text{Density}, \text{Centrality}, \text{ContagionPaths})$

Step 1 — Pairwise causal screening

For all pairs (V_i, V_j) in V , perform rapid causal screening using kernel-based conditional independence testing:

$$\text{CausalEdge}(V_i, V_j) \iff V_i \not\perp V_j \mid Z \quad \forall Z \subseteq V \setminus \{V_i, V_j\}$$

For computational tractability in large enterprise variable sets, use the PC algorithm's skeleton discovery phase with a maximum conditioning set size k_{\max} .

Step 2 — Causal degree computation

For each node $V_i \in V$, compute:

$$\begin{aligned} \text{InDegree}(V_i) &= |\text{PA}_G(V_i)| \\ \text{OutDegree}(V_i) &= |\text{CH}_G(V_i)| \\ \text{TotalDegree}(V_i) &= \text{InDegree}(V_i) + \text{OutDegree}(V_i) \end{aligned}$$

Step 3 — Causal centrality computation

Compute causal betweenness centrality for each node — the proportion of shortest causal paths in the graph that pass through that node:

$$\text{CausalBetweenness}(V_i) = \sum_{s \neq t \neq V_i} \frac{\sigma_{st}(V_i)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest causal paths from s to t and $\sigma_{st}(V_i)$ is the number passing through V_i .

Step 4 — Causal influence radius computation

For each node V_i , compute the maximum causal influence radius — the maximum causal distance its changes propagate through the mesh:

$$\text{InfluenceRadius}(V_i) = \max_{V_j \in V} d_G(V_i, V_j)$$

where $d_G(V_i, V_j)$ is the length of the shortest directed causal path from V_i to V_j .

Step 5 — Contagion path enumeration

For each high-centrality node (top decile by causal betweenness), enumerate all causal paths through which a failure at that node would propagate:

$$\text{ContagionPaths}(V_i) = \{(V_i \rightarrow \dots \rightarrow V_j) : V_j \in \text{Descendants}_G(V_i)\}$$

Rank contagion paths by their estimated causal effect magnitude and path length.

Step 6 — Critical node identification

Identify systemic risk nodes — nodes whose failure would produce the widest causal contagion:

$$\text{SystemicRisk}(V_i) = \text{CausalBetweenness}(V_i) \times \text{InfluenceRadius}(V_i) \times \text{BusinessImpact}(V_i)$$

Step 7 — Density evolution monitoring

Monitor the causal density of the enterprise mesh over time — detecting when new causal connections are forming (increasing systemic risk) or existing connections are weakening (decreasing resilience):

$$\Delta \text{Density}(t) = \text{MeshDensity}(t) - \text{MeshDensity}(t - \Delta t)$$

8.3 Enterprise Applications

Systemic risk monitoring. The CDM produces the first causally grounded systemic risk map — identifying the nodes whose failure would propagate most widely through the enterprise causal mesh and the paths through which contagion would travel.

Merger and acquisition integration risk. When two enterprises merge, their causal meshes combine — creating new causal connections and amplifying existing ones. The CDM maps the causal density of the combined entity and identifies regions of elevated systemic risk created by the merger.

Technology platform dependency analysis. Enterprise technology platforms exhibit mesh causal topology — multiple business functions causally dependent on shared infrastructure. The CDM identifies the causal density of technology dependencies and the systemic risk created by high-centrality platform failures.

Regulatory stress testing. The CDM provides the causal architecture for stress test scenario propagation — identifying the causal paths through which a stress scenario at one node propagates to all affected nodes, and quantifying the total causal impact.

8.4 Output Specification

The CDM produces:

- A causal density heat map of the enterprise causal graph
- A systemic risk ranking of all enterprise nodes by causal betweenness, influence radius, and business impact
- A contagion path catalogue for all high-centrality nodes
- A real-time density evolution monitor tracking mesh dynamics

9. Algorithm 7: Latent Topology Detection — The Hidden Cause Excavator (HCE)

9.1 Definition

Latent Causal Topology is a causal structure in which an unobserved common cause — a hidden variable — generates multiple observed variables simultaneously, creating spurious correlations between observed variables that appear causally connected but are causally independent.

Formal definition: A causal system exhibits latent topology over observed variables $O = \{O_1, \dots, O_k\}$ with hidden cause L if:

- $L \notin O$ — L is unobserved
- $L \in \text{PA}_G(O_i)$ for all i — L causally generates each observed variable
- $O_i \perp_G O_j \mid L$ for all $i \neq j$ — observed variables are conditionally independent given L

- $O_i \not\sim O_j$ in the naive observed-variable causal graph — the variables appear correlated without L

Structural properties:

- Spurious correlations between observed variables
- True causal independence conditional on the latent cause
- Interventions on observed variables do not affect other observed variables
- Only identification and control of the latent cause reveals true causal structure

9.2 The Hidden Cause Excavator Algorithm

Input: Observed data $O = \{O_1, \dots, O_k\}$, candidate proxy variable set Z **Output:** Latent cause registry $LC = \{\hat{L}, \text{proxies}, \text{affected_observables}, \text{estimated_effect}\}$

Step 1 — Spurious correlation identification

Identify pairs of observed variables that are statistically correlated but whose correlation cannot be explained by any observed variable in the current causal graph:

$$\text{SuspectedLatent}(O_i, O_j) \text{ iff } O_i \not\perp O_j \text{ AND } O_i \perp O_j \mid Z = \emptyset$$

These are pairs whose correlation is unexplained by any observed conditioning set — the signature of a latent common cause.

Step 2 — Proxy variable identification

Search the observed variable set for proxy variables — variables that are causally influenced by the hypothesised latent cause but not directly by the target observed variables:

$$\text{Proxy}(Z_k, L, O) \text{ iff } Z_k \not\perp O_i \text{ for all } O_i \in \text{AffectedObservables}(L) \text{ AND } Z_k \perp O_i \mid L \text{ for all } O_i$$

In practice, proxy identification is guided by domain knowledge — for example, in financial risk models, geographic and temporal variables often serve as proxies for latent macroeconomic factors.

Step 3 — Latent variable estimation

Using the identified proxy variables and the proxy variable estimation framework (Miao et al., 2018), estimate the latent common cause \hat{L} :

$$\hat{L} = g(Z, O; \hat{\theta})$$

where g is a function estimated to satisfy the proxy variable identification conditions and $\hat{\theta}$ are the estimated parameters.

Step 4 — Spurious correlation decomposition

Decompose the observed correlations into genuine causal effects and latent common cause effects:

$$\text{Corr}(O_i, O_j) = \underbrace{\text{DirectCausal}(O_i, O_j)}_{\text{genuine}} + \underbrace{\hat{\beta}_i \hat{\beta}_j \text{Var}(\hat{L})}_{\text{latent}} + \underbrace{\epsilon_{ij}}_{\text{noise}}$$

where $\hat{\beta}_i$ is the estimated causal coefficient from \hat{L} to O_i .

Step 5 — Causal graph correction

Remove spurious edges from the causal graph — edges between observed variables that are explained by the latent common cause rather than direct causal relationships. Add the estimated latent variable \hat{L} as a node in the corrected causal graph with edges to all causally affected observed variables.

Step 6 — Latent cause monitoring

Deploy the estimated latent variable as a real-time monitoring signal. Track its inferred value continuously and alert when it reaches levels historically associated with adverse outcomes across its multiple observed effects.

9.3 Enterprise Applications

Hidden market factor identification. The HCE identifies latent macroeconomic factors — business cycle position, credit cycle phase, risk appetite — that drive correlated movements across multiple asset classes, credit quality indicators, and operational metrics simultaneously.

Fraud network detection. The coordinating intent of an organised fraud network is a latent common cause driving apparently independent fraud events across multiple customers, products, and geographies. The HCE identifies the latent network structure from its observed effects.

Organisational dysfunction diagnosis. Leadership failures, cultural misalignment, and incentive dysfunction are latent common causes driving multiple operational failures across apparently unrelated functions. The HCE identifies the latent organisational cause from its diverse operational effects.

Risk model confounder control. Statistical risk models confound genuine causal risk factors with latent common causes — producing risk estimates that are systematically wrong under stress when the latent factor intensifies. The HCE identifies and controls for latent confounders, producing causally grounded risk estimates.

9.4 Output Specification

The HCE produces:

- A latent cause registry: estimated latent variables, their proxy indicators, their affected observables, and their estimated causal effects
- A spurious correlation map: all observed correlations decomposed into genuine causal and latent common cause components
- A corrected causal graph: the enterprise causal graph with spurious edges removed and latent variables added
- Real-time latent cause monitoring signals with alert thresholds

10. Algorithm 8: Interventional Topology Detection — The Control Point Locator (CPL)

10.1 Definition

Interventional Causal Topology is a causal structure that explicitly represents the points in a system where intelligent agents — human or artificial — can apply controlled interventions to override normal causal flow, and the causal consequences of those interventions.

Formal definition: An interventional topology over a causal graph G is an augmented graph $G_I = (V \cup \{A\}, E \cup E_A)$ where A is a set of intervention agents and $E_A \subseteq A \times V$ are intervention edges — directed edges from agents to the variables they can intervene upon — with the property that $\text{do}(V_i = v)$ removes all edges in $\text{PA}_G(V_i)$ and sets $V_i = v$.

Structural properties:

- Explicit representation of intervention points and their permitted agents
- Do-calculus semantics: intervention overrides normal causal flow
- Counterfactual support: interventional topology enables what-if analysis
- Governance embedding: intervention boundaries are structural properties of the topology

10.2 The Control Point Locator Algorithm

Input: Enterprise causal graph G , intervention history H_I , governance framework GF **Output:** Intervention map $IM = (V_I, E_I, \text{Leverage}, \text{Boundaries}, \text{OptimalSequences})$

Step 1 — Intervention history analysis

Identify historical instances of deliberate causal intervention in enterprise processes:

$$H_I = \{(t, V_i, v_i, \text{Context}, \text{Outcome}) : \text{NormalCausalFlow}(V_i, t) \text{ was overridden}\}$$

Interventions include: risk limit triggers, regulatory actions, management decisions, automated controls, emergency procedures.

Step 2 — Intervention effect estimation

For each identified intervention point V_i , estimate the causal effect of intervention using the do-calculus:

$$\hat{\tau}(V_i) = \mathbb{E}[O \mid \text{do}(V_i = v_{\text{intervention}})] - \mathbb{E}[O \mid V_i = v_{\text{counterfactual}}]$$

where O is the relevant outcome variable and $v_{\text{counterfactual}}$ is the value V_i would have taken without intervention, estimated from the unintervened causal dynamics.

Step 3 — Leverage scoring

Score each intervention point by its causal leverage:

$$\text{Leverage}(V_i) = \frac{|\hat{\tau}(V_i)| \times \text{OutcomeImportance}(O)}{\text{InterventionCost}(V_i)}$$

High-leverage intervention points are where intelligent agents can produce maximum causal impact per unit of intervention effort.

Step 4 — Governance boundary mapping

Map the regulatory, risk management, and operational constraints defining when and how intervention at each point is authorised:

$$\text{Boundary}(V_i) = \{(\text{Condition}, \text{AuthorisedIntervention}, \text{AuthorisingAgent}) : \text{GF}\}$$

These boundaries are embedded as structural constraints in the interventional topology — the intervention map is a governance document as well as a causal architecture map.

Step 5 — Optimal intervention sequencing

For multi-step intervention scenarios — achieving a desired causal outcome through a sequence of interventions at multiple points — compute the optimal sequence using counterfactual simulation:

$$\text{OptimalSequence}(O_{\text{target}}) = \arg\max_{\text{sequence} \in \Pi} P(O = O_{\text{target}} \mid \text{do}(\text{sequence}))$$

where Π is the set of all intervention sequences permitted by the governance boundaries.

Step 6 — Agentic AI governance integration

For agentic AI systems operating within the enterprise, embed the interventional topology as the governance architecture:

- The agent identifies available intervention points from the intervention map
- It estimates the causal consequences of candidate interventions using the do-calculus
- It ranks interventions by the SO's generative efficiency criterion
- It executes only interventions within its authorised governance boundaries
- It logs all interventions with their estimated and actual causal effects for audit

10.3 Enterprise Applications

Agentic AI governance. The CPL provides the causal architecture for governing autonomous AI agents in enterprise environments — explicitly representing where agents can act, what the causal consequences of their actions will be, and what governance boundaries constrain their intervention authority.

Regulatory compliance automation. Regulatory requirements are interventional topology structures — points where regulators intervene in normal market and enterprise causal dynamics. The CPL maps these points and their causal consequences, enabling automated compliance monitoring and intervention triggering.

Risk limit management. Risk limits are interventional topology control points — thresholds at which normal risk-taking behaviour is overridden by mandatory position reduction or hedging. The CPL maps all risk limit control points and their causal consequences for portfolio dynamics.

Operational resilience response. Emergency response procedures are interventional topology structures — predefined intervention sequences for restoring normal causal flow when it is disrupted. The CPL maps these sequences and optimises them for causal effectiveness.

10.4 Output Specification

The CPL produces:

- An intervention map: all identified intervention points, their leverage scores, their causal effect estimates, and their governance boundaries
- An optimal intervention sequence library: pre-computed optimal intervention sequences for defined outcome objectives
- An agentic AI governance architecture: the complete interventional topology specification for autonomous agent deployment
- An intervention audit log: all historical and real-time interventions with their estimated and actual causal effects

11. The Meta-Algorithm: The Hybrid Topology Classifier (HTC)

11.1 The Hybrid Reality of Enterprise Causal Systems

No enterprise domain exhibits a single pure causal topology. Every real enterprise system exhibits a hybrid — a simultaneous combination of multiple topologies operating at different levels of abstraction, in different domains, and across different temporal scales. The most consequential enterprise risk events — systemic crises, major fraud incidents, operational catastrophes — arise precisely from the interaction of multiple topologies: a divergent failure triggering a cyclical amplification that propagates through a mesh to produce a convergent catastrophe.

The Hybrid Topology Classifier is the meta-algorithm that integrates the outputs of all eight topology detection algorithms, identifies the simultaneous topology combinations present in each enterprise domain, and maps the cross-topology interactions that produce the most complex and consequential enterprise dynamics.

11.2 The Hybrid Topology Classifier Algorithm

Input: Outputs of all eight topology detection algorithms {CCT, RCC, MFAS, FLI, ALM, CDM, HCE, CPL}, enterprise domain structure D **Output:** Enterprise causal topology signature $\Sigma = (\text{DomainTopologies}, \text{CrossTopologyInteractions}, \text{TemporalDynamics})$

Step 1 — Domain topology inventory

For each enterprise domain $d_k \in D$, compile the topology inventory — the set of causal topologies detected within that domain:

$$\text{Topology}(d_k) = \{T \in \{L, D, C, Cy, H, M, La, I\} : \text{DetectionAlgorithm}(T, d_k) > \theta_T\}$$

where L, D, C, Cy, H, M, La, I denote Linear, Divergent, Convergent, Cyclical, Hierarchical, Mesh, Latent, and Interventional topologies respectively, and θ_T is the detection confidence threshold for each topology type.

Step 2 — Cross-domain topology mapping

Identify causal relationships between topology instances across domains — where a topological structure in one domain causally interacts with a topological structure in another:

$$\text{CrossTopology}(T_i^{d_k}, T_j^{d_l}) \text{ iff } \exists V_i \in T_i^{d_k}, V_j \in T_j^{d_l} : V_i \in \text{PA}_G(V_j) \text{ or } V_j \in \text{PA}_G(V_i)$$

Step 3 — Interaction pattern identification

Identify characteristic cross-topology interaction patterns — topology combinations that produce distinctive enterprise risk dynamics:

Divergent-Cyclical cascade: A divergent topology (model failure causing multiple simultaneous anomalies) feeding into a cyclical topology (the anomalies triggering a feedback amplification loop). This pattern produces the most rapid escalation of enterprise risk.

Latent-Mesh contagion: A latent common cause (hidden macro factor) activating simultaneously across multiple nodes in a mesh topology (interconnected enterprise functions). This pattern produces simultaneous, correlated failures across apparently independent systems.

Convergent-Hierarchical assembly: Multiple independent necessary conditions (convergent topology) assembling at a lower hierarchical level before propagating upward to produce a higher-level catastrophic outcome. This pattern produces the most difficult-to-detect pre-crisis conditions.

Cyclical-Interventional regulation: A positive feedback loop (credit or liquidity cycle) being regulated by intervention topology (central bank policy, regulatory capital requirements). This pattern governs the effectiveness of macroprudential policy.

Step 4 — Temporal dynamics mapping

Map the temporal dynamics of detected topologies and their interactions:

- Which topologies are currently active versus dormant
- Which topologies are evolving — gaining strength, changing phase, extending scope
- Which cross-topology interactions are currently engaged versus latent
- What the temporal trajectory of the enterprise causal topology signature is

Step 5 — Enterprise causal topology signature generation

Generate the enterprise causal topology signature — a comprehensive, structured representation of the organisation's causal architecture:

$$\Sigma = \{ \{ (d_k, \text{Topology}(d_k)) \}_{k=1}^{|D|}, \text{CrossTopologyInteractions}, \text{TemporalDynamics}, \text{SystemicRiskProfile} \}$$

The signature is the deepest possible characterisation of the enterprise's causal architecture — more informative than any statistical risk metric, more predictive than any historical performance analysis, more actionable than any process map.

Step 6 — Signature evolution monitoring

Monitor the enterprise causal topology signature continuously — detecting when it evolves, when new topologies emerge, when existing topologies change character, and when cross-topology interactions develop or dissolve.

Alert when the signature evolves toward configurations historically associated with adverse outcomes — the approach of a dangerous convergent assembly, the strengthening of a positive feedback loop, the emergence of a new latent common cause.

11.3 The Enterprise Causal Topology Signature

The enterprise causal topology signature is the unique fingerprint of an organisation's causal architecture. It characterises:

Domain topology profile: Which topologies are present in which domains, at what intensity, and with what temporal stability.

Cross-topology interaction map: How topological structures in different domains causally interact — the pathways through which a topological event in one domain propagates to another.

Systemic risk profile: The regions of the enterprise causal architecture where topology combinations create systemic risk — where a failure in one topology could cascade through cross-topology interactions to produce enterprise-wide consequences.

Resilience profile: The regions of the enterprise causal architecture where topology combinations create natural resilience — where negative feedback loops, redundant causal paths, or high-leverage intervention points provide natural buffering against adverse causal cascades.

Temporal dynamics: How the enterprise causal topology signature evolves over time — whether the organisation's causal architecture is becoming more complex, more fragile, more resilient, or more opaque.

No two enterprises have the same causal topology signature. The signature reflects the specific combination of business model, organisational structure, technology architecture, market position, and regulatory context that makes each enterprise causally unique.

11.4 Output Specification

The HTC produces:

- The enterprise causal topology signature — the comprehensive characterisation of the organisation's causal architecture
- A cross-topology interaction map — the causal pathways through which topological structures in different domains interact
- A systemic risk profile — the topology combinations that create systemic risk
- A resilience profile — the topology combinations that create natural resilience
- A signature evolution monitor — continuous tracking of how the enterprise causal architecture changes over time

12. Implementation Architecture

12.1 GHA Implementation

The nine causal topology detection algorithms are implemented within the GHA's self-extending causal world model. The GHA provides:

Causal graph substrate: The directed causal graph maintained by the GHA provides the structural substrate on which all nine algorithms operate. Each algorithm reads from and writes to this shared causal graph — reading detected causal structures and writing confirmed topology instances.

Self-extension for novel topologies: When a detection algorithm identifies a causal mechanism that the current GHA architecture cannot represent — a new feedback loop, a previously undetected latent cause, a new intervention point — it triggers GHA self-extension. The GHA autonomously extends its architecture to represent the new mechanism, integrating it into the existing causal world model.

Cross-algorithm consistency: The GHA's unified causal graph ensures consistency across the outputs of all nine detection algorithms. A causal edge confirmed by the CCT is immediately available to the CDM and the HTC — the causal knowledge accumulated by each algorithm enriches the operating context of all others.

12.2 SO Integration

The Sustainability Optimizer directs the computational resource allocation of the detection framework by generative efficiency:

Detection priority: The SO allocates detection effort — the frequency and depth of each algorithm's execution — in proportion to the expected causal richness of the topology being detected relative to the computational cost of detection.

Alert calibration: The SO calibrates detection thresholds to maximise the generative efficiency of the alert stream — maximising the causal significance of generated alerts relative to the operational cost of alert handling.

Extension direction: When GHA self-extension is triggered by a detection algorithm, the SO evaluates candidate extension directions by generative efficiency — selecting the architectural extension that will produce the greatest causal understanding relative to its computational cost.

12.3 Deployment Architecture

The detection framework is deployed as a layered architecture:

Layer 1 — Event ingestion: Enterprise events from all data sources are ingested through a streaming pipeline and routed to the relevant detection algorithms based on domain and variable type.

Layer 2 — Topology detection: The eight topology-specific algorithms execute continuously on ingested data, updating their respective outputs in real time.

Layer 3 — Hybrid classification: The HTC integrates the outputs of all eight algorithms continuously, maintaining and updating the enterprise causal topology signature.

Layer 4 — Alert and insight generation: Alerts, intervention recommendations, and causal insights are generated from the HTC output and routed to the appropriate enterprise stakeholders.

Layer 5 — Governance and audit: All causal inferences, topology detections, and alert generations are logged with full causal provenance for regulatory audit and model governance purposes.

13. Limitations and Future Directions

13.1 Computational Scalability

The causal topology detection algorithms, particularly the CDM's pairwise causal screening and the MFAS's factor combination testing, face computational scalability challenges as the number of enterprise variables grows. Future work should develop:

- Scalable approximate causal discovery methods for large variable sets
- Hierarchical causal screening that exploits the enterprise's natural modular structure
- Distributed causal graph processing for enterprise-scale deployment

13.2 Identifiability Under Partial Observability

Several algorithms — particularly the HCE — require strong identifiability conditions that may not hold in all enterprise contexts. Future work should extend the framework to:

- Partial identifiability results for latent topology detection under weaker conditions
- Sensitivity analysis for detection algorithm outputs under identifiability violations
- Robust detection methods that maintain validity under partial observability

13.3 Dynamic Topology Evolution

Enterprise causal topologies evolve over time — new topologies emerge, existing ones change character, cross-topology interactions develop and dissolve. The current framework detects static topologies and monitors their evolution. Future work should develop:

- Causal topology change point detection — identifying when topology transitions occur
- Predictive topology evolution modelling — forecasting future topology configurations
- Topology lifecycle management — guiding enterprises through topology transitions

13.4 Hybrid Topology Interaction Dynamics

The cross-topology interaction patterns identified by the HTC are catalogued but not yet fully formalised. Future work should develop:

- A formal theory of cross-topology interaction dynamics
- Mathematical characterisation of dangerous topology combinations
- Optimal intervention strategies for cross-topology risk scenarios

14. Conclusion

This paper has introduced Causal Topology Detection — a systematic framework of nine algorithms for identifying and monitoring the fundamental causal architecture of enterprise systems. The framework identifies eight fundamental causal topologies — Linear, Divergent, Convergent, Cyclical, Hierarchical, Mesh, Latent, and Interventional — and a meta-algorithm, the Hybrid Topology Classifier, that characterises the unique causal topology signature of each enterprise.

The nine algorithms — the Causal Chain Tracer, Root Cause Crystalliser, Multi-Factor Assembly Scanner, Feedback Loop Identifier, Abstraction Level Mapper, Causal Density Mapper, Hidden Cause Excavator, Control Point Locator, and Hybrid Topology Classifier — constitute the first systematic framework for causal topology detection in enterprise systems. Implemented through VectorPeak's

Generative Hierarchy Algorithm and Sustainability Optimizer, they enable AI that understands the generative architecture of organisations rather than merely approximating their statistical surface.

The enterprise causal topology signature — the unique fingerprint of an organisation's causal architecture — is the deepest possible characterisation of enterprise risk and resilience properties. It is what the GHA discovers, what the SO uses to direct causal reasoning resources, and what distinguishes genuine enterprise causal intelligence from the statistical approximations that have governed enterprise AI until now.

Enterprise organisations are causal systems of extraordinary complexity. The framework introduced in this paper gives them, for the first time, AI that understands that complexity at the level of its generative structure.

The causal architecture of the enterprise was always there. Now we have the algorithms to see it.

References

- Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.
- Peters, J., Janzing, D., & Schölkopf, B. (2017). *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press.
- Spirtes, P., Glymour, C., & Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press.
- Chickering, D. M. (2002). Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3, 507–554.
- Zheng, X., Aragam, B., Ravikumar, P., & Xing, E. P. (2018). DAGs with NO TEARS: Continuous Optimization for Structure Learning. *NeurIPS 2018*.
- Zhang, K., Peters, J., Janzing, D., & Schölkopf, B. (2011). Kernel-Based Conditional Independence Test and Application in Causal Discovery. *UAI 2011*.
- Miao, W., Geng, Z., & Tchetgen Tchetgen, E. J. (2018). Identifying Causal Effects with Proxy Variables of an Unmeasured Confounder. *Biometrika*, 105(4), 987–993.
- Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., & Schölkopf, B. (2009). Nonlinear Causal Discovery with Additive Noise Models. *NeurIPS 2008*.
- Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., & Bengio, Y. (2021). Toward Causal Representation Learning. *Proceedings of the IEEE*, 109(5), 612–634.
- Granger, C. W. J. (1969). Investigating Causal Relations by Econometric Models and Cross-Spectral Methods. *Econometrica*, 37(3), 424–438.
- Bareinboim, E., & Pearl, J. (2016). Causal Inference and the Data-Fusion Problem. *Proceedings of the National Academy of Sciences*, 113(27), 7345–7352.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. *ICLR 2018*.
- Prigogine, I., & Stengers, I. (1984). *Order Out of Chaos: Man's New Dialogue with Nature*. Bantam Books.

[Author]. (2026). The Meta Generative Principle. *Zenodo*. <https://doi.org/10.5281/zenodo.19549127>

[Author]. (2026). GHA and SO Algorithms — MGP Computational Implementation. *Zenodo*.
<https://doi.org/10.5281/zenodo.19582200>

[Author]. (2026). The Causal Imperative. *Zenodo*. <https://doi.org/10.5281/zenodo.19615588>

[Author]. (2026). The Inevitable Mind. *Zenodo*. <https://doi.org/10.5281/zenodo.19616056>

[Author]. (2026). The Generative Ontology. *Zenodo*. <https://doi.org/10.5281/zenodo.19616056>

Corresponding author: Vallikat Peethamber, VectorPeak is a technology consulting and advisory firm specialising in causal AI architecture and deployment for regulated industries. Published open access under CC BY 4.0 Cite as: Vallikat Peethamber (2026). Causal Topology Detection: A Framework of Algorithms for Enterprise Causal Intelligence. VectorPeak Applied Frameworks, No. 1. Zenodo.
