Concept Paper

The G Model: A Geometric Approach to Absolute Data Coherence in Information Systems

José Vicente Quiles Feliu *

Posted Date: 13 January 2026

doi: 10.20944/preprints202601.0490.v2

Keywords: database theory; data coherence; geometric data models; formal verification; information systems architecture

 doi:10.20944/preprints202601.0490.v2

Concept Paper

The G Model: A Geometric Approach to Absolute Data Coherence in Information Systems

Josep Vicent Quiles Feliu

Independent Researcher, Spain; jvquilesf@gmail.com

Abstract

Modern information systems suffer from a fundamental architectural flaw: data coherence depends on external validation layers, creating systemic entropy and computational waste. We present the **G Model**, a mathematical framework that redefines information as points in a geometric space where incoherence is *mathematically impossible*. Through a triaxial formalization (Meaning, Location, Connection) and an intrinsic coherence operator (Φ), the system guarantees that only valid data can exist within the managed universe (Ω). We formalize this with four fundamental axioms ensuring coherence, uniqueness, acyclicity, and deterministic propagation. The model extends Codd's normalization through five semantic normal forms addressing temporal and semantic coherence. We provide formal proofs of key theorems, including optimal propagation complexity and impossibility of inconsistency. This work represents a paradigm shift from "data storage systems" to "coherent information spaces," providing a foundation for trustworthy AI training data and critical infrastructure where error is inadmissible.

Keywords: database theory; data coherence; geometric data models; formal verification; information systems architecture

---

1. Introduction

1.1. Motivation

For the past five decades, data management has been anchored in the "passive container" paradigm. Both Relational Database Management Systems (RDBMS) following Codd's model [1] and their modern derivatives (NoSQL, NewSQL) operate under a problematic separation: storage is the database's responsibility, while logical coherence is delegated to external application layers.

This fragmented approach generates what we term **Systemic Entropy**:

- **External Validation:** Data integrity depends on fallible, mutable middleware external to the data itself
- **Coherence Latency:** A temporal gap exists between data mutation and consequence propagation
- **Structural Fragility:** Relationships (Foreign Keys) are mere pointers, not strict causal dependencies
- **Computational Waste:** Continuous validation requiring redundant processing

Current approaches attempt resolution through increased computing power and software patches (triggers, complex stored procedures). However, this builds on fundamentally flawed foundations: the underlying architecture continues treating information as "static cells" rather than dynamic vectors with geometric properties.

1.2. Research Questions

This work addresses three fundamental theoretical questions:

 doi:10.20944/preprints202601.0490.v2

RQ1: Can data coherence be guaranteed *mathematically* rather than validated *procedurally*?

RQ2: What formal framework enables intrinsic coherence while maintaining computational tractability?

RQ3: What are the theoretical properties and limitations of such a framework?

1.3. Contributions

Our main theoretical contributions are:

1. **The G Model:** A formal geometric framework for information spaces where incoherence is mathematically impossible (Section 3)
2. **Four Fundamental Axioms:** Formal guarantees of existence, uniqueness, acyclicity, and propagation determinism (Section 3.3)
3. **SRGD Normal Forms:** Five semantic normal forms extending Codd's work to temporal and semantic domains (Section 3.5)
4. **Propagation Theorems:** Proofs of optimal complexity and impossibility of inconsistency (Section 4)
5. **Architectural Principles:** Abstract requirements for systems implementing the G Model (Section 5)
6. **Observer Coherence Algebra:** Formal model guaranteeing coherence at observation boundaries (Section 6)

1.4. Paper Organization

Section 2 reviews related work. Section 3 formalizes the G Model. Section 5 derives architectural principles. Section 6 presents observer coherence algebra. Section 7 analyzes implications and limitations. Section 8 concludes. Appendix A contains formal proofs.

2. Related Work

2.1. Traditional Database Theory

Codd's Relational Model (1970) [1] established modern RDBMS foundations through set theory and first-order predicate logic. The model defines relations as subsets of Cartesian products and introduces normalization (1NF-5NF) to eliminate redundancies. However, Codd's model treats coherence as *external constraints* (PRIMARY KEY, FOREIGN KEY, CHECK) enforced by the DBMS engine or application logic.

Fundamental limitation: Coherence is reactive—validated after attempted violation. Complex business rules require external triggers or procedures with no formal propagation model for calculated values.

Date's refinements [2] maintain this separation between data and coherence logic.

2.2. NoSQL and Graph Databases

NoSQL systems (MongoDB, Cassandra, DynamoDB) sacrifice ACID guarantees for scalability, implementing eventual consistency [3]. This exacerbates coherence problems by explicitly allowing temporary incoherent states.

Graph databases (Neo4j, ArangoDB) [4] model relationships as first-class entities, similar to our F axis. However:

- Coherence still requires external validation
- No native distinction between base and calculated values
- No formal acyclicity guarantees in the data model

 doi:10.20944/preprints202601.0490.v2

2.3. Constraint Programming and Formal Methods

Alloy [5] and Z notation [6] provide formal specification languages for system invariants. These are *specification tools*, not execution models—they verify designs but don't implement self-coherent structures.

Active Databases [7] introduced ECA (Event-Condition-Action) rules embedding logic in databases. However, triggers remain imperative constructs, not declarative mathematical guarantees.

2.4. Type Systems and Dependent Types

Dependent type systems [8] (Coq, Agda, Idris) allow types to depend on values, enabling compile-time correctness proofs. Our Φ operator shares philosophical similarities but operates at the *data level* rather than *program level*, providing runtime guarantees in production systems.

2.5. Gaps in Literature

No existing formal framework provides:

1. Intrinsic coherence where invalid data *cannot exist* (not just "shouldn't")
2. Geometric formalization of information with topological properties
3. Deterministic propagation distinguishing base from calculated data
4. Formal algebra extending coherence guarantees across system boundaries
5. Unified theoretical treatment of multi-channel information flows

The G Model addresses these gaps through a novel geometric interpretation of information spaces.

3. The G Model: Formal Framework

3.1. Global Information Space

We define the **Global Information Space** as a set of points g determined by a triaxial vector:

$$G = \{A, K, F\} \tag{1}$$

$$g = (a, k, f) \quad \text{where} \quad a \in A, k \in K, f \in F \tag{2}$$

The three axes represent:

- A **(Meaning Axis):** Attribute defining data semantics, governed by norms $\mathcal{N} = \{\mathcal{N}_{\text{Dom}}, \mathcal{N}_{\text{Stx}}, \mathcal{N}_{\text{Lim}}\}$
  - $\mathcal{N}_{\text{Dom}}$: Domain constraints (type, range)
  - $\mathcal{N}_{\text{Stx}}$: Syntactic constraints (format, pattern)
  - $\mathcal{N}_{\text{Lim}}$: Limit constraints (business rules)
- K **(Location Axis):** Unique identity key in the universe, $K \subset \mathbb{N}$
- F **(Connection Axis):** Foreign keys establishing directed graph topology, $F \subset K$

Definition 1 (Information Point)**.** *An information point is a tuple with an associated value:*

$$\forall g \in G : g = (a, k, f) \land \exists v \in \mathcal{V}, v = Val(g)$$

where $\mathcal{V}$ is the value domain appropriate to attribute a.

3.2. The Coherent Management Universe

Unlike traditional systems permitting inconsistent transient states, the **real management universe** Ω is defined exclusively as the subset of G where the Coherence Operator equals unity.

 doi:10.20944/preprints202601.0490.v2

Definition 2 (Coherent Universe)**.**

$$\Omega = \{g \in G \mid \Phi_{\mathcal{N}}(g) = 1\}$$

Definition 3 (Coherence Operator)**.**

$$\Phi_{\mathcal{N}} : G \to \{0, 1\}$$

For each point $g = (a, k, f)$, the operator evaluates compliance with all applicable norms:

$$\varphi(g) = \varphi_{Dom}(Val(g)) \wedge \varphi_{Stx}(Val(g)) \wedge \varphi_{Lim}(Val(g))$$

where each φ_i is a predicate evaluating the corresponding norm.

Property 1 (Impossibility of Incoherence)**.**

$$\forall g \in G : g \notin \Omega \implies g \text{ cannot be stored in the system}$$

This is not validation that rejects; it is a *geometric property* preventing existence. Points failing $\Phi_{\mathcal{N}} = 1$ are not members of Ω by definition.

3.3. Fundamental Axioms

The G Model is governed by four irreducible axioms:

Axiom 1 (Existence by Coherence)**.**

$$\forall g \in G : g \in \Omega \iff \Phi_{\mathcal{N}}(g) = 1$$

Implication: The "erroneous datum" does not exist within Ω. Error is an external anomaly failing to crystallize in the coherent universe.

Axiom 2 (Uniqueness of Location)**.**

$$\forall k \in K, k \in R \implies \forall k' \in R, (k' \equiv k \implies k' = k)$$

where R is any relation and $\equiv$ denotes semantic equivalence.

Implication: The location axis is orthogonal and unique; there is no positional ambiguity.

Axiom 3 (Direction and Non-Circularity)**.** *The dependency relation $\prec$ on Ω satisfies:*

1. ***Irreflexivity:*** $\forall g \in \Omega : g \not\prec g$
2. ***Asymmetry:*** $\forall g, g' \in \Omega : (g \prec g') \implies (g' \not\prec g)$
3. ***Acyclic Transitivity:*** $\forall g, g', g'' \in \Omega : (g \prec g' \wedge g' \prec g'') \implies (g'' \not\prec g)$

Implication: The dependency graph (V, E) where $V = \Omega$ and $E = \{(g, g') \mid g \prec g'\}$ is necessarily a DAG (Directed Acyclic Graph), guaranteeing algorithm termination and system stability.

 doi:10.20944/preprints202601.0490.v2

Axiom 4 (Determinism of Propagation)**.**

$$\forall g \in \Omega : \delta(g) \in \{0, 1\}$$

where:

- $\delta(g) = 1 \implies g \in G_{base}$ *(source of information, independently observable)*
- $\delta(g) = 0 \implies g \in G_{calc}$ *(dependent on propagation vector* Π*)*

Implication: The system has complete knowledge of dependency structure, enabling identification of exactly which points are affected by changes to base data.

3.4. Propagation Dynamics

Definition 4 (Dependency Graph)**.** *For a point with location* k_0*, the dependency graph is:*

$$H(k_0) = \left\{ g \in \Omega \mid (k_0, k(g)) \in \bigcup_{n \geq 0} F^n \right\}$$

where F^n *denotes the n-fold composition of the connection relation.*

This captures all points reachable from k_0 through the foreign key topology.

Definition 5 (Propagation Vector)**.** *For a base point* $g_0 \in G_{base}$*:*

$$\Pi(g_0) = \left\{ g \in H(k_0) \cap G_{calc} \mid (g_0, g) \in \bigcup_{n \geq 1} \prec^n \right\}$$

The propagation vector identifies exactly those calculated points whose values depend (directly or transitively) on g_0.

Theorem 1 (Optimal Propagation)**.** *When a base datum* g_0 *changes, only points in* $\Pi(g_0)$ *are affected, with dependency identification complexity:*

$$O(|\Pi(g_0)|) \ll O(|\Omega|)$$

Proof. See Appendix A.1. The proof establishes that $\Pi(g_0)$ precisely identifies all calculated points whose values depend on g_0, and that identifying this set has complexity proportional to $|\Pi(g_0)|$ rather than $|\Omega|$. □

Corollary 1 (Propagation Locality)**.** *For typical dependency structures:*

$$|\Pi(g_0)| \ll |H(k_0)| \ll |\Omega|$$

ensuring surgical identification of affected values rather than global recomputation.

3.5. SRGD Normal Forms

While Codd's normal forms (1NF-5NF) eliminate syntactic redundancies, **SRGD Normal Forms** extend into semantic and temporal domains.

 doi:10.20944/preprints202601.0490.v2

Definition 6 (SRGD-FN1: Semantic Atomic Grouping). *A relation R is in SRGD-FN1 if:*

$$FN1(R) \iff R \in 5NF_{classic} \land (\forall a \in Attr(R), \forall k \in K : \exists Val(a,k)) \land \exists E : Attr(R) \subseteq Sem(E)$$

where:

- $5NF_{classic}$ *denotes Codd's fifth normal form*
- *The second condition ensures no NULL values (total functions)*
- $Sem(E)$ *denotes the semantic closure of entity* E

Interpretation: Each relation represents a semantically complete entity with unitary meaning. Attributes form a minimal sufficient set capturing the entity's essence without nullability.

Definition 7 (SRGD-FN2: Systemic Unification). *Relations R_i, R_j satisfy FN2 if:*

$$Sem(R_i) = Sem(R_j) \implies \exists R : (R_i, R_j) \rightsquigarrow R$$

where $\rightsquigarrow$ denotes semantic unification.

Interpretation: Semantically identical entities unify into a single systemic structure, eliminating conceptual redundancy beyond syntactic duplication.

Definition 8 (SRGD-FN3: Semantic Hierarchization). *Relations organize hierarchically:*

$$R_{base}(A_0, K) \prec R_{spec}(A_0 \cup A_s, K)$$

where A_0 are common attributes and A_s are specialization attributes.

Interpretation: Entity hierarchies capture generalization-specialization relationships, with upper categories encapsulating shared properties and lower categories adding specific attributes.

Definition 9 (SRGD-FN4: Relational Role Emergence). *For an entity e:*

$$Role(e) = \Phi(Rel(e))$$

where $Rel(e)$ is the set of active relationships.

Interpretation: Roles are not static attributes but emerge dynamically from relationship states. An entity's role is a function of its position in the relational graph topology.

Definition 10 (SRGD-FN5: Temporal Semantic Coherence). *Value evolution satisfies:*

$$Val_t(g) \xrightarrow{\Pi,\delta} Val_{t+1}(g)$$

governed by:

$$Val_{t+1}(g) = \begin{cases} Val_t(g) & \text{if } \delta(g) = 1 \land \neg Modified(g) \\ f(\Pi(g), t) & \text{if } \delta(g) = 0 \end{cases}$$

 doi:10.20944/preprints202601.0490.v2

Interpretation: Temporal evolution is deterministic. Base values change only through explicit modification; calculated values propagate according to dependency structure, guaranteeing coherence across time without imperative intervention.

4. Theoretical Properties

4.1. Coherence Theorems

Theorem 2 (Global Coherence Invariant)**.** *At any system state:*

$$\forall g \in \Omega : \Phi_{\mathcal{N}}(g) = 1$$

Proof. By Definition 2, membership in Ω is equivalent to $\Phi_{\mathcal{N}}(g) = 1$. Therefore, the invariant holds by construction. □

Theorem 3 (Propagation Correctness)**.** *After modification of base point g_0, when calculated points in $\Pi(g_0)$ are evaluated:*

$$\forall g \in \Pi(g_0) : Val'(g) = f_g(\{Val'(g') \mid g' \in Dep(g)\})$$

where f_g is the calculation function and $Dep(g) = \{g' \mid g' \prec g\}$.

Proof. See Appendix A.2. The proof demonstrates that evaluating calculated points in topological order ensures correctness regardless of when evaluation occurs. □

Theorem 4 (Impossibility of Inconsistency)**.** *It is impossible for the system to contain inconsistent data:*

$$\neg\exists g \in \Omega : \Phi_{\mathcal{N}}(g) = 0$$

Proof. Suppose for contradiction that $\exists g \in \Omega$ with $\Phi_{\mathcal{N}}(g) = 0$. By Definition 2, $g \in \Omega \implies \Phi_{\mathcal{N}}(g) = 1$. This contradicts our supposition. Therefore, no such g exists. □

4.2. Complexity Analysis

Lemma 1 (DAG Property)**.** *The dependency graph $(\Omega, \prec)$ forms a DAG.*

Proof. By Axiom 3, $\prec$ is irreflexive, asymmetric, and transitively acyclic. These properties are precisely the definition of a DAG. □

Theorem 5 (Dependency Identification Complexity)**.** *Identifying $\Pi(g_0)$ (the set of calculated points affected by change to g_0) has complexity:*

$$O(|V| + |E|)$$

where $V = H(k_0)$ and E are edges in the dependency subgraph, with:

$$|V| + |E| \ll |\Omega|$$

for typical topologies.

Proof. See Appendix A.3. □

 doi:10.20944/preprints202601.0490.v2

Corollary 2 (Sublinear Average Case)**.** *For randomly distributed dependencies:*

$$\mathbb{E}[|\Pi(g_0)|] = O(\log |\Omega|)$$

4.3. Topological Properties

Theorem 6 (Compactness of Ω)**.** *Under the discrete topology, Ω is compact.*

Proof. $\Omega \subset G$ is defined by a finite set of constraints $\mathcal{N}$. Each constraint defines a closed set. Ω is the intersection of finitely many closed sets, hence closed. Being a subset of the discrete space G, it is also bounded, thus compact. □

Theorem 7 (Continuity of Φ)**.** *The coherence operator $\Phi_{\mathcal{N}} : G \to \{0, 1\}$ is continuous under appropriate metric on G.*

Proof. Define metric $d(g_1, g_2) = 0$ if $g_1 = g_2$, else $d(g_1, g_2) = 1$. Under this discrete metric, every function is continuous. For more meaningful topologies (e.g., value-based), Φ remains continuous as it's defined by finite Boolean combinations of continuous predicates φ_i. □

5. Architectural Principles

While the G Model is purely mathematical, any system implementing it must satisfy certain architectural principles. We derive these abstractly without specifying technologies.

5.1. Statelessness Principle

Definition 11 (Stateless Communication)**.** *A system is stateless if each request-response cycle is independent:*

$$\forall r_1, r_2 \in \mathcal{R} : Process(r_2) \not\to State(r_1)$$

where $\mathcal{R}$ is the set of requests and $State(r)$ is any residual state from processing r.

Theorem 8 (Stateless Scalability)**.** *Stateless systems exhibit perfect horizontal scalability:*

$$Throughput(n \cdot S) = n \cdot Throughput(S)$$

where S is a server instance and n is the number of instances.

Proof. Without shared state, each request is independently routable to any instance. Load distribution is trivial (round-robin, random, least-loaded), and adding instances increases capacity linearly without coordination overhead. □

5.2. Tripartite Architecture

Any system materializing the G Model naturally decomposes into three abstract layers:

1. **Persistence Layer:** Stores and retrieves data at the logic layer's request. Performs no autonomous processing.
2. **Logic Layer:** Adapts information flows between the persistence layer and observation layer according to $\mathcal{N}$ and the specific flow type.
3. **Observation Layer:** Manages external interaction

 doi:10.20944/preprints202601.0490.v2

- Renders interfaces from server-provided specifications
- Presents fields (both $\delta = 1$ and $\delta = 0$) to observers
- Allows editing according to norms $\mathcal{N}$
- Validates locally via Φ (aggregation of field-level φ)
- Transmits to server only if $\Phi = 1$
- Blocks transmission if $\Phi = 0$

Key principle: Logic layer is thin—complexity resides in $\mathcal{N}$ (declarative metadata) rather than imperative code.

5.3. Unified Flow Pattern

Definition 12 (Information Flow)**.** *An information flow is a 4-tuple:*

$$\mathcal{F} = (\mathcal{N}_{config}, \Phi_{validation}, \mathcal{D}_{channel}, \mathcal{F}_{format})$$

where:

- $\mathcal{N}_{config}$*: Applicable norms from repository*
- $\Phi_{validation}$*: Coherence operator configuration*
- $\mathcal{D}_{channel}$*: Data channel (human interface, API, sensor, etc.)*
- $\mathcal{F}_{format}$*: Representation format*

Axiom 5 (Flow Orthogonality)**.** *For distinct flows $\mathcal{F}_i, \mathcal{F}_j$:*

$$i \neq j \implies \mathcal{F}_i \cap \mathcal{F}_j = \emptyset$$

Activity in one flow does not affect latency or resource availability of others.

Implication: All information sources—human users, IoT sensors, external APIs, AI systems—are treated uniformly. Each validates against $\mathcal{N}$, applies Φ, and updates Ω through identical mechanisms. This enables channel-agnostic coherence guarantees.

6. Observer Coherence Algebra

External observers (users, APIs, displays) interact with Ω through an interface layer requiring its own coherence model. We formalize this as the **RM/O Model** (Relational Model for Observers).

6.1. Observer Objects

Definition 13 (Observer Object)**.** *An observer object is a 4-tuple:*

$$o = (A_h, \mathcal{M}, \mathcal{E}, \Sigma)$$

where:

- $A_h \subseteq H(k_0)$*: Projected attributes from dependency graph*
- $\mathcal{M} = \{M_{syn}, M_{dom}\}$*: Validation methods*
- $\mathcal{E}$*: Operational state (visible, enabled, editable)*
- Σ*: Event signals (change, focus, blur)*

Each object encapsulates a fragment of Ω with local coherence validation.

 doi:10.20944/preprints202601.0490.v2

6.2. Local and Global Coherence

Definition 14 (Local Coherence Operator)**.** *For observer object o:*

$$\varphi(o) = M_{syn}(Val(o)) \wedge M_{dom}(Val(o))$$

where M_{syn} validates syntax and M_{dom} validates domain constraints.

Definition 15 (Container)**.** *A container C is a special observer object aggregating other objects:*

$$C = \{o_1, o_2, \ldots, o_n\}$$

Definition 16 (Global Coherence Operator)**.** *For container C:*

$$\Phi(C) = \bigwedge_{o_i \in C} \varphi(o_i)$$

6.3. Transfer to Ω

Definition 17 (Transfer Morphism)**.**

$$\tau : \mathcal{O} \to \Omega$$

where $\mathcal{O}$ is the space of observer objects.

Axiom 6 (Atomicity Principle)**.** *Transfer executes atomically only if container is globally coherent:*

$$\tau(C) = \begin{cases} \{\tau(o_1), \tau(o_2), \ldots, \tau(o_n)\} & \text{if } \Phi(C) = 1 \\ \varnothing & \text{if } \Phi(C) = 0 \end{cases}$$

Interpretation: All-or-nothing semantics. If a single field within a form is incoherent, the entire transfer is blocked. No partial updates reach Ω.

6.4. Formal Properties

Invariant 1 (Binary Coherence)**.**

$$\forall o \in \mathcal{O} : \varphi(o) \in \{0, 1\}$$

Invariant 2 (Coherence Aggregation)**.**

$$\forall C \in \mathcal{C} : \Phi(C) = \bigwedge_{o_i \in C} \varphi(o_i)$$

where $\mathcal{C}$ is the set of containers.

Invariant 3 (Object Autonomy)**.**

$$\forall o_1, o_2 \in \mathcal{O} : \nexists f : o_1 \to o_2$$

Observer objects are independent; coherence is local, not relational.

 doi:10.20944/preprints202601.0490.v2

Invariant 4 (Transfer Atomicity)**.**

$$\tau(o) \neq \bot \implies \Phi(C_{parent}(o)) = 1$$

where $C_{parent}(o)$ *is the container holding* o*.*

Theorem 9 (Transitive Coherence)**.**

$$\forall C_1, C_2 \in \mathcal{C} : C_2 \subset C_1 \wedge \Phi(C_1) = 1 \implies \Phi(C_2) = 1$$

Proof. Suppose $\Phi(C_1) = 1$. By Definition 16, $\bigwedge_{o \in C_1} \varphi(o) = 1$, so $\varphi(o) = 1$ for all $o \in C_1$. Since $C_2 \subset C_1$, all $o \in C_2$ satisfy $\varphi(o) = 1$, thus $\Phi(C_2) = \bigwedge_{o \in C_2} \varphi(o) = 1$. □

Theorem 10 (Impossibility of Partial Inconsistency)**.**

$$\Phi(C) = 0 \implies \forall o \in C : \tau(o) = \bot$$

Proof. By Axiom 6, $\tau(C) = \emptyset$ when $\Phi(C) = 0$. This means no object within C transfers to Ω, hence $\tau(o) = \bot$ for all $o \in C$. □

7. Discussion

7.1. Theoretical Implications

7.1.1. Paradigm Shift

The G Model represents a fundamental reconceptualization: from *data storage systems* to *coherent information spaces*. Coherence is not a goal to achieve through validation but an inherent, inescapable property of the space itself.

Traditional systems ask: "How do we ensure this data is valid?"

The G Model asks: "What space contains only valid data?" and then constructs that space mathematically.

7.1.2. Relationship to Type Theory

The $\Phi_{\mathcal{N}}$ operator shares philosophical kinship with dependent types [8]. Both make correctness a precondition for existence. However:

- **Dependent types:** Operate at *compile-time* on *program terms*
- **G Model Φ:** Operates at *runtime* on *data values*

This enables runtime guarantees in production systems where data arrives from external sources (users, sensors, APIs) that cannot be type-checked at compile-time.

7.1.3. Topological Perspective

By treating information as a geometric space, we inherit mathematical properties:

- $H(k_0)$ as reachability set in directed graph
- Ω as filtered subspace of G
- $\Pi(g_0)$ as forward closure under $\prec$
- Coherence operator Φ as characteristic function of Ω

 doi:10.20944/preprints202601.0490.v2

This geometric view enables application of graph algorithms, topological analysis, and categorical constructions to data management problems.

7.2. Practical Advantages

While this paper focuses on theory, we note several practical benefits emerging from the formal framework:

1. **Stateless Architecture:** Theorem 8 guarantees perfect horizontal scalability
2. **Surgical Dependency Identification:** Theorem 1 ensures identifying affected values has complexity proportional to dependency size, not system size
3. **On-Demand Evaluation:** Calculated values (where $\delta(g) = 0$) are evaluated when requested, ensuring always-current results from base data
4. **AI Training Data:** Clean-by-design Ω provides trustworthy corpus for machine learning
5. **Formal Auditability:** All coherence logic resides in declarative $\mathcal{N}$, enabling automated verification

7.3. Limitations and Open Problems

7.3.1. Computational Complexity

While $|\Pi(g_0)| \ll |\Omega|$ typically, pathological cases exist:

- **Deeply nested hierarchies:** Computing $H(k_0)$ approaches $O(|V| + |E|)$
- **High fan-out:** Base point affecting thousands of calculated points
- **Dense dependency graphs:** Approaching $O(n^2)$ edges

Open problem: Develop incremental algorithms for H computation exploiting locality and caching.

7.3.2. Distributed Ω

Current formalization assumes single-authority Ω. Extending to distributed settings while maintaining Axiom 1 presents challenges:

- **Consensus:** How to achieve $\Phi_{\mathcal{N}}(g) = 1$ across nodes with potential network partitions?
- **Eventual coherence:** Can we relax $\Phi = 1$ to "eventually $\Phi = 1$" while preserving useful properties?
- **Partitioned $\mathcal{N}$:** Different nodes enforcing different norms

Open problem: Formalize distributed G Model with CAP theorem tradeoffs explicit.

7.3.3. Temporal Queries

SRGD-FN5 addresses temporal coherence but doesn't provide full temporal query algebra. Questions like "What was Val(g) at time t?" or "When did g enter Ω?" require additional formalization.

Open problem: Integrate temporal database theory [9] with G Model, defining $\Omega(t)$ as time-varying coherent space.

7.3.4. Probabilistic Coherence

The binary $\Phi : G \to \{0, 1\}$ is rigid. Real systems may benefit from probabilistic coherence:

$$\Phi_p : G \to [0, 1]$$

where $\Phi_p(g) = 0.95$ means "95% confident this datum is coherent."

Open problem: Extend G Model to fuzzy or probabilistic coherence while maintaining useful guarantees.

 doi:10.20944/preprints202601.0490.v2

7.4. Future Research Directions

1. **Formal Verification:** Machine-check Axioms 1–4 and Theorems 1–10 using Coq or Isabelle
2. **Benchmark Development:** Create standard test suite comparing G Model implementations against traditional RDBMS, NoSQL, and graph databases across:
   - Coherence violation rates
   - Propagation efficiency
   - Scalability curves
   - Query performance
3. **Machine Learning Integration:** Investigate using Ω as training corpus for LLMs, measuring:
   - Reduction in hallucination rates
   - Improvement in factual accuracy
   - Explainability through $H(k_0)$ lineage
4. **Category Theory Formalization:** Explore categorical interpretation:
   - Ω as object in category of coherent spaces
   - Φ as natural transformation
   - Transfer τ as functor from observer category to Ω category
5. **Quantum Extension:** Investigate quantum coherence operators where superposition of states maintains $\Phi = 1$ for all components

7.5. Comparison with Related Formalisms

Only the G Model provides intrinsic coherence, deterministic propagation, formal acyclicity guarantees, and extends coherence algebra to system boundaries.

Table 1. Comparison of formal properties across approaches.

Property	RDBMS	Graph DB	Type Systems	G Model
Coherence	Reactive	Reactive	Compile-time	Intrinsic
Propagation	None/Trigger	None	N/A	Deterministic (Π)
Acyclicity	Not guaranteed	Not guaranteed	Guaranteed	Guaranteed (Axiom 3)
Base/Calc	Not distinguished	Not distinguished	Not applicable	Formal (δ)
Stateless	No	No	N/A	Provable (Theorem 8)
Observer Algebra	Ad-hoc	Ad-hoc	N/A	Formal (RM/O)

8. Conclusions

We presented the **G Model**, a mathematical framework redefining information as points in a geometric space where incoherence is impossible by construction. Through four fundamental axioms (Existence by Coherence, Uniqueness of Location, Direction and Non-Circularity, Determinism of Propagation), we formalize coherence, uniqueness, acyclicity, and deterministic propagation.

8.1. Key Theoretical Results

1. **Impossibility of Incoherence (Theorem 4):** Invalid data cannot exist in Ω, not merely "shouldn't exist"
2. **Optimal Dependency Identification (Theorem 1):** Changes affect only points in $\Pi(g_0)$ with identification complexity $O(|\Pi|) \ll O(|\Omega|)$

 doi:10.20944/preprints202601.0490.v2

3. **Stateless Scalability (Theorem 8):** Perfect horizontal scaling with $\text{Throughput}(n \cdot S) = n \cdot \text{Throughput}(S)$
4. **Observer Coherence (Theorems 9, 10):** Formal guarantees extend across system boundaries through RM/O algebra
5. **SRGD Normal Forms:** Five semantic extensions (FN1-FN5) of Codd's work addressing temporal and semantic coherence

8.2. Research Questions Answered

RQ1 (Mathematical Coherence): Yes, through the $\Phi_{\mathcal{N}}$ operator and Definition 2 of Ω as filtered subspace

RQ2 (Efficient Framework): Geometric formalization with Φ, Π, δ enables $O(|\Pi|)$ dependency identification and stateless architecture

RQ3 (Theoretical Properties): Formal proofs establish correctness, efficiency bounds, and limitations

8.3. Paradigm Shift

The G Model transforms data management from *storage systems with external validation* to *coherent information spaces with intrinsic guarantees*. This makes yesterday's impossible errors mathematically impossible today.

8.4. Impact and Applications

While this paper focuses on theoretical foundations, the framework enables:

- **Trustworthy AI:** Clean training data from Ω reduces hallucinations
- **Critical Infrastructure:** Mathematical coherence guarantees for finance, energy, defense
- **Formal Verification:** Declarative $\mathcal{N}$ enables automated correctness proofs
- **Scalable Systems:** Stateless architecture natural to cloud-native deployments

8.5. Future Directions

Open problems include distributed Ω formalization, integration with temporal databases, probabilistic coherence extensions, and machine verification using proof assistants.

The G Model provides a rigorous mathematical foundation for next-generation information systems where coherence is not a goal but an inescapable property of the space itself.

Acknowledgments: The author thanks the anonymous reviewers for their insightful feedback and suggestions that significantly improved this work.

Appendix A. Formal Proofs

Appendix A.1. Proof of Theorem 1 (Optimal Propagation)

Theorem A1 (Restatement). *When base datum g_0 where $\delta(g_0) = 1$ changes, only points in propagation vector $\Pi(g_0)$ are affected, with dependency identification complexity $O(|\Pi(g_0)|) \ll O(|\Omega|)$.*

Proof. 1. By Definition 5,

$$\Pi(g_0) = \left\{ g \in H(k_0) \cap G_{\text{calc}} \mid (g_0, g) \in \bigcup_{n \geq 1} \prec^n \right\}$$

 doi:10.20944/preprints202601.0490.v2

2. This set contains exactly the calculated points reachable from g_0 through the dependency relation $\prec$.
3. By Axiom 3, the dependency graph is acyclic. Therefore, $\prec$ defines a partial order.
4. Consider any point $g \notin \Pi(g_0)$. Two cases:
   (a) $\delta(g) = 1$: Point g is base, doesn't depend on anything
   (b) g not reachable from g_0 through $\prec$: No dependency path connects g_0 to g
5. In both cases, changing $\text{Val}(g_0)$ cannot affect $\text{Val}(g)$ since there's no dependency path.
6. Therefore, only $g \in \Pi(g_0)$ will have different values when subsequently evaluated.
7. Since dependency graphs typically have bounded fan-out and depth:

$$|\Pi(g_0)| \leq d \cdot f^d \ll |\Omega|$$

   where d is maximum depth and f is average fan-out.
8. Computing $\Pi(g_0)$ requires traversing the DAG from g_0, which is $O(|V_\Pi| + |E_\Pi|)$ where V_Π and E_Π are vertices and edges in the propagation subgraph.
9. Since $|V_\Pi| = |\Pi(g_0)| \ll |\Omega|$ and edges are typically sparse, complexity is:

$$O(|\Pi(g_0)|) \ll O(|\Omega|) \quad \square$$

Appendix A.2. Proof of Theorem 3 (Propagation Correctness)

Theorem A2 (Restatement). *When calculated points in $\Pi(g_0)$ are evaluated after modification of base point g_0:*

$$\forall g \in \Pi(g_0) : Val'(g) = f_g(\{Val'(g') \mid g' \in Dep(g)\})$$

Proof. 1. By Lemma 1, the dependency graph is a DAG.
2. Perform topological sort on $\Pi(g_0)$, yielding ordering $g_1, g_2, \ldots, g_n$ where:

$$g_i \prec g_j \implies i < j$$

3. Process points in this order when evaluating. For each g_i:
   (a) All $g' \in \text{Dep}(g_i)$ appear earlier in the ordering (by topological sort)
   (b) Therefore, $\text{Val}'(g')$ is available for all dependencies (either base values or already evaluated)
   (c) Evaluate: $\text{Val}'(g_i) = f_{g_i}(\{\text{Val}'(g') \mid g' \in \text{Dep}(g_i)\})$
4. By induction:
   - **Base case:** g_0 is modified explicitly, so $\text{Val}'(g_0)$ is correct
   - **Inductive step:** If all dependencies of g_i have correct values, then $\text{Val}'(g_i) = f_{g_i}(\ldots)$ is correct by definition of f_{g_i}
5. Therefore, all points in $\Pi(g_0)$ yield correct values when evaluated in topological order.

□

Appendix A.3. Proof of Theorem 5 (Complexity Bound)

Theorem A3 (Restatement). *Identifying $\Pi(g_0)$ (the set of calculated points affected by change to g_0) has complexity $O(|V| + |E|)$ where $V = H(k_0)$ and E are edges in the dependency subgraph.*

Proof. 1. **Computing $H(k_0)$:** Breadth-first search from k_0 through foreign key graph F:
   - Visit each reachable vertex once: $O(|H(k_0)|)$

 doi:10.20944/preprints202601.0490.v2

- Examine each edge once: $O(|\{(k, k') \mid k \in H(k_0), k' \in F(k)\}|)$
- Total: $O(|V_H| + |E_H|)$ where subscript H denotes restriction to $H(k_0)$

2. **Filtering to $\Pi(g_0)$:** For each $g \in H(k_0)$:
   - Check $\delta(g) = 0$ (calculated): $O(1)$
   - Check reachability from g_0 through $\prec$: Already known from BFS
   - Total: $O(|H(k_0)|)$
3. **Topological sort of $\Pi(g_0)$:** Standard algorithm:
   - Compute in-degrees: $O(|V_\Pi| + |E_\Pi|)$
   - Process queue: $O(|V_\Pi|)$
   - Total: $O(|V_\Pi| + |E_\Pi|)$
4. **Evaluation (if performed):** Evaluating all $g \in \Pi(g_0)$ in topological order:
   - Each g evaluation depends on $|\text{Dep}(g)|$ points
   - Each dependency examined once across all evaluations: $O(|E_\Pi|)$
   - Total: $O(|V_\Pi| + |E_\Pi|)$
5. **Overall complexity:**

$$O(|V_H| + |E_H|) + O(|V_\Pi| + |E_\Pi|) = O(|V_H| + |E_H|)$$

   since $\Pi(g_0) \subseteq H(k_0)$.
6. For typical dependency structures:
   - Average depth: $\bar{d} = O(\log |\Omega|)$
   - Average fan-out: $\bar{f} = O(1)$ (constant)
   - Thus $|H(k_0)| = O(\bar{f}^{\bar{d}}) = O(\log |\Omega|)$
7. Therefore: $O(|V| + |E|) \ll O(|\Omega|)$ for typical topologies.

□

References

1. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.
2. C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston: Addison-Wesley, 2003.
3. G. DeCandia et al., "Dynamo: Amazon's Highly Available Key-value Store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.
4. R. Angles and C. Gutierrez, "Survey of Graph Database Models," *ACM Computing Surveys*, vol. 40, no. 1, pp. 1–39, 2008.
5. D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. Cambridge: MIT Press, 2012.
6. J. M. Spivey, *The Z Notation: A Reference Manual*, 2nd ed. Upper Saddle River: Prentice Hall, 1992.
7. N. W. Paton and O. Díaz, "Active Database Systems," *ACM Computing Surveys*, vol. 31, no. 1, pp. 63–103, 1999.
8. B. C. Pierce, *Types and Programming Languages*. Cambridge: MIT Press, 2002.
9. C. S. Jensen et al., "The Consensus Glossary of Temporal Database Concepts," *ACM SIGMOD Record*, vol. 23, no. 1, pp. 52–64, 1994.