TITLE

Causonomy: A Formal Science of Failure and Causation in Normative Systems

By Pascal Etcheber

Independent Researcher, United Kingdom

April 2026

ABSTRACT

Problem solving is universally practised yet lacks a formal scientific foundation. Existing approaches rely on heuristic search, domain expertise, or probabilistic inference because failure has not been defined as a closed and enumerable object. This paper introduces Causonomy, a science in which failure is defined as a measurable deviation between a required state $R(f)$and an observed state $O(f)$at an activity boundary, such that:

$$D(f) = R(f) - O(f), D(f) \neq 0$$

and formalised as a Negative Outcome:

$$NO = (a, \delta)$$

Starting from this definition, the paper derives a finite ontology of failure. Forms and activities are governed by primitive dimensions—structure and quantity at the level of form, and existence, magnitude, and time at the level of activity—from which a complete grammar of twelve deviation types is obtained. Combined with six universal activity functions, these yield a closed space of seventy-two Negative Outcomes, constituting the complete surface of observable failure.

Causation is located at the governing level. Required states are specified through governing forms structured by governing functions, requirement primitives, and lifecycle states. Structural defects within this governing layer constitute root causes. The governing-form universe is finite, and the relation between governing defects and observable failures is encoded in a Structural Reduction Matrix:

$$SRM(g, no) = 1 \iff Projection(g, no) \wedge Connectivity(g, no)$$

which determines admissibility a priori through identity perturbation and structural reachability within the system. As a consequence, the space of admissible causes for any failure is known in advance and is finite. The framework yields a complete grammar of twelve deviation types, a closed space of seventy-two Negative Outcomes, eight

hundred and sixty-four operational causal positions, and a governing-form universe of 1,152 root-cause positions at the governing level.

The framework supports four operations—diagnosis, prediction, stress testing, and preventive design—each defined as a traversal of the same structural system. Diagnosis maps a failure to its admissible causes; prediction maps a governing defect to its admissible consequences; stress testing enumerates all failures permitted by a system's governing structure; preventive design modifies that structure to remove admissible failures. In each case, reasoning proceeds by elimination within a closed space rather than by hypothesis generation.

The results establish problem solving as a domain governed by necessary relations rather than interpretive methods. Completeness of causal enumeration follows from closure of the failure and cause spaces, reducing dependence on expertise and eliminating first-order causal selection. The framework is falsifiable: it would be invalidated by any failure, cause, or admissible relation outside the defined spaces. The work defines a new science with a finite object, necessary relations, and explicit conditions of refutation, transforming problem solving from a heuristic practice into a deterministic discipline.

Keywords: Causonomy; failure ontology; root cause analysis; causal admissibility; Structural Reduction Matrix; normative systems

Table of Contents

1. Introduction — The Missing Science

Problem solving occupies a central place in organised activity, yet it remains structurally undefined. Organisations depend on it to operate, improve, and survive; entire disciplines have formed around diagnosing, correcting, and preventing failure. Despite that centrality, no formal science governs the activity itself. Methods exist in abundance—root cause analysis, fault trees, statistical models, expert systems—but none rests on a closed and necessary description of the object they seek to analyse. As a consequence, completeness cannot be guaranteed. Causal explanations remain contingent, dependent on expertise, interpretation, and the limits of enumeration.

The absence of a science does not reflect a lack of effort or sophistication. It reflects a deeper omission. Scientific domains emerge when their objects are defined in a manner that admits necessity and closure[1]. Geometry became possible when space was no longer described empirically but structured axiomatically. Information theory emerged when information was bounded and measurable rather than contextual and semantic[2]. In each case, a field crossed from method to science when an object that had appeared indefinitely variable revealed a finite structure governed by law. Problem solving has not undergone this transition because failure itself has never been isolated as such an object.

Failure appears everywhere yet remains conceptually diffuse. In practice, it is described through domain language: defects in manufacturing, incidents in software, shortages in supply chains, errors in medicine, delays in logistics. Each description carries the vocabulary and assumptions of its domain. Causes are expressed as narratives—what happened, who acted, what changed—rather than as positions within a structured space. As explanations accumulate, the space of possible causes appears to expand without bound. Under those conditions, problem solving becomes an exercise in exploration. Brainstorming, hypothesis generation, and expert judgement replace deduction because no finite domain constrains admissibility.

Such an approach has practical limits. When the space of possible causes is undefined, completeness cannot be demonstrated. Different analysts may propose different explanations without contradiction. Evidence is used to support preferred interpretations rather than to eliminate impossibilities. As systems grow in scale and complexity, the number of plausible narratives increases, and the act of explanation becomes slower, more costly, and less certain. Recurrent problems persist not because they are inherently complex, but because the space in which they are analysed remains open.

---

[1] Kant (1781/1998)
[2] Shannon (1948)

A science of problem solving requires a different foundation. The first requirement is not a method but an object. Failure must be defined independently of domain, interpretation, and narrative. That definition must satisfy two conditions. First, it must capture all instances of failure without remainder. Second, it must admit a finite structure. Without these conditions, no complete reasoning system can exist. With them, causation becomes a matter of necessity rather than conjecture.

The central claim of this paper is that such a definition exists. Failure in normative systems can be expressed as the deviation of an observed state from a required state at a defined boundary of activity. Once expressed in that form, failure admits a finite grammar. The ways in which a state can deviate are limited. The ways in which work can produce or transform states are likewise limited. Their combination yields a closed set of possible failures. What appears as endless variation reduces to a finite and enumerable space.

From that closure follows a second result. If the space of failures is finite, the space of their causes must also be finite. Causes do not arise arbitrarily; they arise from the failure of conditions necessary to produce required states. Those conditions can be defined independently of domain and enumerated exhaustively. Causation therefore belongs to a closed universe, and the relation between causes and failures can be determined in advance. The need for hypothesis generation disappears. Problem solving becomes an exercise in locating a position within a known structure and eliminating alternatives through evidence.

The framework developed in this paper establishes that structure. It begins by defining the object of the science—Negative Outcome—as the measurable deviation between required and observed states. It then derives a finite set of primitive properties of form and a complete grammar of deviation. Combined with a closed set of activity types, these yield the complete space of observable failure. Causation is then located at the governing level, where structural defects occur within a finite universe of requirement primitives and lifecycle states. A Structural Reduction Matrix encodes the admissible relation between governing defects and observable outcomes, while a graph representation of system structure defines the propagation of those effects.

The result is a unified system of reasoning with four operations: diagnosis, prediction, stress testing, and preventive design. Each operation traverses the same structure in a different direction. Diagnosis maps observed failures to admissible causes. Prediction maps known structural defects to their possible consequences. Stress testing evaluates the full space of potential failures implied by a system's governing structure. Preventive design modifies that structure to eliminate or constrain admissible failures. No new machinery is introduced across these operations. The same ontology, mappings, and constraints apply throughout.

The implications are both theoretical and practical. Theoretically, the work establishes problem solving as a science defined by a closed object, finite domain, and necessary relations. Practically, it replaces exploratory reasoning with deterministic elimination. Once a failure is correctly stated and the system is completely described, the full set of admissible causes is known in advance. Evidence no longer supports conjecture; it removes impossibilities. Completeness is guaranteed by structure, not by expertise.

The claim advanced here is precise and falsifiable[3]. The science would be invalidated if a failure were observed that could not be expressed as a deviation within the proposed grammar, or if a cause were found outside the enumerated causal space that nevertheless produced such a failure. The absence of such cases is not assumed but follows from the derivations that follow.

Problem solving has long been practised as a craft supported by experience, intuition, and tools. The work presented here shows that it can be grounded in a formal structure. Once that structure is made explicit, the activity ceases to depend on exploration and becomes an application of necessity.

2. The Object of the Science — Negative Outcome

A science requires a clearly defined object. Without such an object, reasoning cannot be constrained, and completeness cannot be established. Problem solving has historically proceeded without this foundation because failure has been treated as a contextual or narrative phenomenon rather than as a formal entity. The first step in establishing a science of problem solving is therefore to define failure in a way that is independent of domain language and admits precise observation.

Failure arises in systems that operate under requirements. A requirement specifies what must be true of a form at a given point in a system. Let F denote the set of forms, where a form is any entity upon which activity operates or whose state is governed. For each form $f \in F$, a required state $R(f)$ is defined by the governing structure of the system. The required state may concern identity, quantity, structure, or any property necessary for correct operation. The required state is not inferred after the fact; it is established prior to execution through specifications, plans, constraints, or rules.

Execution produces an observed state. Let $O(f)$ denote the observed state of form f at the boundary of an activity. The observed state is the result of all prior transformations, transfers, and governing influences acting upon the form. The system operates correctly when the observed state satisfies the requirement, that is, when:

$$R(f) = O(f)$$

[3] Popper (1959)

Failure appears when this equality does not hold.

The difference between required and observed states defines deviation. Formally, deviation is expressed as:

$$D(f) = R(f) - O(f)$$

A deviation exists if and only if:

$$D(f) \neq 0$$

The expression $R(f) - O(f)$does not imply numerical subtraction in all cases. It denotes a generalised difference across the properties that define the required state. A structural mismatch, a missing element, a delay, an excess quantity, or a deficient magnitude all correspond to non-zero values of $D(f)$under appropriate interpretation. The essential point is that deviation is defined as a relation between two states, not as an isolated attribute of one.

A problem is defined as the presence of such a deviation. More precisely, a problem is the failure of a supported expectation. An expectation is supported when it is grounded in governing structure: it is specified, authorised, and intended by the system. Failure is therefore not merely any difference, but a difference relative to a requirement that has legitimacy within the system. This distinction excludes incidental variation and confines the object of the science to deviations that matter for system performance.

The notion of failure remains incomplete until it is located within the structure of activity. Systems do not operate in a continuous, undifferentiated manner; they consist of discrete activities that transform, transfer, or verify forms. Each activity defines a boundary at which the state of a form is produced, received, or assessed. It is at these boundaries that required and observed states can be meaningfully compared. A deviation cannot be assigned to a system in general; it must be associated with a specific activity boundary at which the expectation applies.

Let Adenote the set of activities. A Negative Outcome is defined as the occurrence of a deviation at a specific activity:

$$NO = (a, \delta), a \in A,\ \delta \in \Delta$$

A Negative Outcome is therefore an activity-bound deviation. It specifies both where the failure is observed and how the observed state differs from the required state. The definition removes ambiguity by anchoring failure at the precise boundary where expectation and execution are compared.

Such formulation resolves a persistent difficulty in problem solving: the conflation of symptoms with failures. A delay observed by a customer is not itself the failure; the failure lies at the activity boundary where the required delivery condition was not met. A shortage observed at the point of use is not defined by absence alone but by the failure of a prior activity to produce or supply the required quantity. Binding failure to activity boundaries ensures that each Negative Outcome corresponds to a specific, structurally identifiable point of comparison.

The definition also establishes that failure is observable and measurable. Because both $R(f)$and $O(f)$are defined states, the deviation $D(f)$can be determined without recourse to interpretation. The measurement may be numerical, categorical, or logical, depending on the nature of the requirement, but it remains grounded in a comparison of explicitly defined states. Such property is essential for scientific treatment. An object that cannot be observed or measured cannot be systematically analysed.

The concept of Negative Outcome thus provides the formal object required for the science. It satisfies three conditions. First, it is universal: any failure in any domain can be expressed as a deviation between required and observed states at an activity boundary. Second, it is precise: it specifies both the location and the nature of the failure. Third, it is measurable: the deviation can be determined by comparison of defined states.

A critical consequence follows. If all failures can be expressed as activity-bound deviations, then the space of possible failures is determined by the possible activities and the possible forms of deviation. If both sets are finite, the space of Negative Outcomes is finite. The remainder of the paper derives these sets and establishes their closure. The combination of activities and deviations yields a complete and finite enumeration of failure.

The transition achieved in this section is foundational. Failure, previously treated as an open-ended and context-dependent phenomenon, is recast as a structured and bounded object. Once expressed as a measurable gap between required and observed states at a defined boundary, it becomes amenable to formal analysis. The science that follows rests entirely on this definition.

3. The Primitive Ontology of Form

The definition of Negative Outcome established that failure is observed as a deviation between a required state and an observed state at the boundary of an activity. Such a definition presupposes that states themselves are structured in a way that allows comparison. A science of failure therefore requires a prior determination of the dimensions along which forms can differ. Without such determination, deviation remains open-ended, and closure of the failure space cannot be achieved.

It can be shown that, at the level of form, only two primitive dimensions are required: **structure** and **quantity**. These are not empirical attributes collected from observation, but necessary dimensions derived from the conditions under which forms can be governed and evaluated within a system.

A form is any entity upon which an activity operates or whose state is subject to requirement. Forms include physical objects, information artefacts[4], signals, authorisations, instructions, and any structured entity that can be stored, moved, acquired, released, transformed, or checked. The science does not distinguish forms by domain but by their role in governed activity.

For a form to be governed, it must first be identifiable. Structure expresses what the form is. It defines identity, composition, arrangement, and functional characteristics. A requirement always specifies a form in structural terms: a particular component, a specific document, a defined dataset, a valid authorisation, or a correctly configured object. When the observed form does not match what is required, the deviation is structural. The form may be of the wrong type, misaligned with its intended role, incomplete in its composition, inconsistent in its internal relations, unstable over time, or insufficient in its functional capability. All such failures belong to the domain of structure.

Structure is therefore a primitive dimension because no requirement can be expressed without it. A system cannot determine correctness without distinguishing between acceptable and unacceptable forms. The identity of the form is not an optional attribute but a necessary condition for governance. Without structure, there is no basis for comparison between required and observed states.

A second dimension is required because forms are not only identifiable but also present in multiplicity. Quantity expresses how many instances of a form exist relative to what is required. A system may require ten units and observe eight; it may require one authorisation and observe two; it may require continuous availability and observe intermittent presence. These are not failures of identity but failures of count. A form may be structurally correct yet insufficient or excessive in number. Quantity therefore captures a dimension of variation that cannot be reduced to structure.

The distinction between structure and quantity is necessary. A form may be correct in identity but incorrect in number, or incorrect in identity but correct in number. The two dimensions are independent. A shortage is not a structural defect, and a misclassification is not a quantitative defect. Both are failures, but they arise along different axes of variation. Any complete description of a form's state must therefore specify both what the form is and how many instances are present.

[4] Gangemi et al. (2002)

3.1 Criterion for Primitive Identification

The preceding analysis establishes that structure and quantity are candidates for primitive dimensions of form-level deviation. A general rule is required to determine whether such candidates are indeed primitive, and to prevent the introduction of additional dimensions on an ad hoc basis.

The present section defines that rule.

Definition — Primitive Deviation Dimension

A deviation dimension is primitive if and only if it satisfies three conditions simultaneously:

1. **Logical Independence**
2. **Universal Applicability**
3. **Irreducibility under Negation**

These conditions define the admissibility of a dimension within the deviation grammar. Failure to satisfy any one of them disqualifies the candidate.

Logical Independence

A dimension is logically independent if it cannot be derived from, or expressed as a combination of, other dimensions.

Independence is tested by construction. A dimension is independent if a case can be exhibited in which all other dimensions are satisfied, yet a deviation persists along the candidate dimension.

Removal of an independent dimension must therefore leave a class of deviations that cannot be expressed within the remaining set.

Universal Applicability

A dimension is universal if it applies to the specification of forms in any normative system, regardless of domain.

A candidate that applies only to particular classes of systems—physical, biological, informational, or legal—cannot be primitive. A primitive must be meaningful wherever requirements can be defined.

Universality is tested by considering whether the dimension remains necessary across all domains in which forms and requirements exist.

Irreducibility under Negation

A dimension is irreducible if its failure mode cannot be expressed as the failure of another dimension without loss of information.

Negation provides the test. If the statement “the requirement is not satisfied along this dimension” can be reformulated entirely in terms of other dimensions, then the dimension is not primitive.

Irreducibility ensures that each primitive contributes distinct informational content to the description of deviation.

Selection Rule

The three conditions together constitute a formal selection rule:

A deviation dimension is admitted as primitive if and only if it is logically independent, universally applicable, and irreducible under negation.

This rule converts the identification of primitives from a process of illustrative reduction into a deductive procedure. Any proposed dimension must satisfy all three conditions. Any failure to do so establishes that the dimension is derived.

Role in the Argument

The reductions that follow—covering quality, duplication, omission, location, and measurable properties—are not independent demonstrations. Each is an application of the criterion defined above.

The closure of the deviation basis depends on this rule. Without it, the argument remains inductive, relying on the absence of counterexamples. With it, the argument becomes deductive: any proposed additional dimension must satisfy the criterion, and no such dimension can do so.

Forward Reference

The same criterion applies to activity-level primitives. The identification of time, existence, and magnitude in the analysis of activities satisfies the conditions established in this section. Section 4 will refer to this criterion without restating it.

The necessity of these two primitives can be demonstrated by considering the conditions under which a requirement can be satisfied. A requirement must specify the identity of the form it governs; otherwise, it cannot determine whether the correct form is present. It must also specify the required number of instances; otherwise, it cannot determine sufficiency or excess. Remove structure, and the requirement cannot distinguish correct from incorrect forms. Remove quantity, and it cannot distinguish enough from not enough. In both cases, the requirement becomes incapable of guiding action or evaluating outcome. Structure and quantity are therefore necessary.

The sufficiency of the pair follows from reduction. Any apparent additional dimension of form must either be reducible to structure or quantity or belong to the domain of activity rather than form. Consider common candidates. Quality, when understood as correctness of composition or specification, reduces to structure. Quality, when understood as level or performance, reduces either to magnitude of activity or to structural sufficiency of the form. Duplication reduces to quantity excess. Omission reduces to quantity insufficiency if some instances exist, or to activity non-occurrence if none exist. Location, when treated as a property of the form, is a structural relation within the system and therefore belongs to structure. No candidate introduces a third independent dimension of form.

3.2 Continuous Variables and Measurable Properties

Continuous scalar variables are central to many normative systems. Temperature, pressure, voltage, flow rate, concentration, and similar quantities are routinely specified in requirements and measured in operation. A natural objection arises: deviation from such variables appears to define a distinct class of failure, one not captured by structure or quantity.

The objection must be addressed directly.

The Objection

A requirement may specify that a form must satisfy a scalar condition:

- a fluid must be at $320°C$,
- a vessel must operate at 5 bar,
- a signal must remain within a voltage range.

Deviation from such requirements does not immediately appear structural or quantitative. The form may be of the correct type and present in the correct number, yet still fail by virtue of a scalar value. A critic may therefore propose that scalar variables introduce an additional primitive dimension of deviation.

If correct, the deviation basis at the form level would be incomplete.

The Response

Resolution requires application of the primitive identification criterion established in Section 3.1. A candidate dimension must satisfy logical independence, universal applicability, and irreducibility under negation. Continuous variables do not satisfy these conditions.

Scalar Variables as Form Properties

A scalar requirement specifies a property that the form must possess. Temperature, pressure, or voltage are not independent dimensions of deviation; they are specifications of what the form must be.

Failure of such a requirement means the form does not conform to its specification. The deviation concerns the constitution or state of the form.

Such failure is structural:

- absence or deficiency of required property → **S6 (defective)**
- inconsistency with specification → **S4 (inconsistent)**

The scalar value determines how the deviation is measured, but not its type.

Separation of Type and Metric

A distinction is required between:

- **type of deviation**, and
- **metric of deviation**.

The deviation grammar classifies types. Measurement quantifies deviations along those types.

The scalar nature of a variable determines the metric by which deviation is measured, not the type of deviation.

Temperature, pressure, and similar variables provide the scale on which deviation is expressed. They do not define independent axes of deviation.

Failure of the Primitive Criteria

Continuous variables fail the three conditions of Section 3.1:

- **Logical independence**: deviation in scalar value is fully expressible as failure of the form to possess a required property.
- **Universal applicability**: such variables apply only within specific domains.
- **Irreducibility under negation**: the negation of a scalar requirement is equivalent to structural non-conformance.

Failure on any one condition is sufficient. Continuous variables fail all three.

Consequence

No additional primitive dimension arises from measurable properties. The form-level deviation basis remains:

$$\{S, Q\}$$

complete and irreducible.

The role of scalar variables is confined to measurement. They quantify deviation but do not classify it.

3.3 Asymmetry Between Quantity and Scalar Properties

Resolution of the scalar variable objection gives rise to a further question. If temperature, pressure, and similar variables reduce to structure, why does quantity not do the same? Both are measurable. Both can deviate above or below a required value. A symmetry objection therefore arises: either both are primitive, or neither is.

The present section resolves that asymmetry.

Proposition

Quantity is a primitive deviation dimension. Scalar properties such as temperature or pressure are not. The distinction follows from the criterion for primitive identification established in Section 3.1.

Application of the Criterion

A candidate dimension must satisfy:

- logical independence,
- universal applicability,
- irreducibility under negation.

Each condition is applied in turn.

Logical Independence

A form may satisfy all structural requirements—correct type, composition, internal consistency, and properties—and yet be present in an incorrect number.

A set of components may be perfectly specified in every respect and still be insufficient or excessive. No structural deficiency exists in any individual instance. The deviation lies in multiplicity.

Such a case cannot be expressed as a structural deviation. Quantity therefore captures a class of failures not reducible to structure.

Scalar properties do not satisfy this condition. A deviation in temperature or pressure is a failure of the form to possess a required property. The deviation is fully expressible as structural non-conformance.

Universal Applicability

Quantity applies to all forms in all normative systems. Any requirement may specify how many instances are required or may leave multiplicity unconstrained. The possibility of deviation in number is therefore universal.

Scalar properties are domain-specific. Temperature applies to thermodynamic systems, pressure to fluids, voltage to electrical systems. Many normative systems—legal, informational, organisational—do not involve such variables.

Scalar properties therefore fail the universality condition.

Irreducibility under Negation

The negation of a quantity requirement produces deviations such as shortage or excess. These cannot be restated as structural, temporal, or existential failures without loss of information. A shortage is not a wrong type, not a mistimed occurrence, and not an absence of activity. It is specifically a failure of count.

The informational content of quantity deviation is therefore irreducible.

The negation of a scalar requirement, by contrast, is fully expressible as structural non-conformance. A form at the wrong temperature is a form that does not satisfy its specification. No additional information is required beyond that structural failure.

Conclusion

Quantity satisfies all three conditions of the primitive identification criterion. Scalar properties satisfy none.

The asymmetry is therefore not incidental. It follows from the logical structure of requirement specification:

- quantity defines multiplicity of instances, which is independent of what each instance is,
- scalar properties define characteristics of instances, which are part of what the form is.

Consequence

Primitives are not defined by measurability or by the presence of numerical values. They are defined by independence, universality, and irreducibility.

Quantity is primitive because it introduces a dimension of deviation that cannot be eliminated without loss. Scalar variables do not introduce such a dimension.

The deviation basis at the form level remains:

$$\{S, Q\}$$

complete and minimal.

A critical clarification follows. Certain types of deviation—those related to time, existence, and magnitude—are often informally attributed to forms but do not belong to the ontology of form itself. A form may be late, but lateness is not an intrinsic property of the form; it is a property of the activity that failed to produce or deliver the form at the required time. A form may be missing, but absence is not a structural or quantitative attribute of an existing form; it reflects the non-occurrence of an activity that should have produced or transferred it. A form may be insufficient in capacity, but that insufficiency arises from the magnitude at which an activity was performed. These dimensions—time, existence, and magnitude—belong to activities and will be treated as such in the next section. Their apparent association with forms arises from observing the consequences of activity failure at the boundary where forms are evaluated.

The ontology of form is therefore strictly limited to structure and quantity. Structural deviations capture all failures in what the form is. Quantitative deviations capture all failures in how many instances exist. Together, they define the complete space of form-level variation. This result is not empirical but necessary. It follows from the conditions under which forms can be specified, governed, and evaluated.

The significance of this conclusion is foundational. It establishes that the variation of forms is not open-ended. Any failure observed in a form must belong to one of these two dimensions. This constraint is what makes closure possible. Once the ontology of form is fixed, the remaining dimensions of failure can be located in the ontology of activity, and the complete grammar of deviation can be derived.

The science proceeds from this point by integrating the ontology of form with the ontology of activity. The next section will establish the primitive dimensions of activity—time, existence, and magnitude—and show how they complement structure and quantity to produce the full deviation grammar.

4. The Primitive Ontology of Activity

The ontology of form established that all deviations intrinsic to forms belong to two dimensions: structure and quantity. Yet not all failures observed at activity boundaries can be reduced to defects in what a form is or how many instances are present. A form may be correct in identity and sufficient in number yet still fail to satisfy the requirement. Such failures arise not from the form itself but from the manner in which activities are executed. The science therefore requires a second ontology: the primitive dimensions of activity.

An activity is that which produces, transforms, transfers, or verifies the state of a form. Activities operate on forms but are not reducible to them. They are events or processes through which forms come into existence, change, or move across the system. The ontology of activity must therefore capture the fundamental ways in which the execution of an activity can fail relative to its requirement.

It can be shown that exactly three primitive dimensions are required at the level of activity: **time**, **existence**, and **magnitude**. These dimensions correspond to the conditions under which an activity can produce a required outcome. They answer three irreducible questions: *when does the activity occur*, *does it occur at all*, and *at what level does it occur*.

Time is the first primitive because all activity unfolds within a temporal structure. A requirement does not merely specify that an activity should occur; it specifies when it should occur relative to other activities and to system needs. An activity that occurs too early or too late fails to satisfy its requirement even if it produces the correct form in the correct quantity. A delivery that arrives after it is needed is as ineffective as one that does not arrive at all. Time therefore captures the alignment between the occurrence of an activity and the temporal conditions under which its output is required.

Existence is the second primitive because an activity may fail by not occurring. A required transformation may not take place; a transfer may not be initiated; a verification may not be performed. In such cases, the expected form does not appear, not because it is structurally defective or insufficient in quantity, but because the activity that should have produced it did not occur. Existence therefore captures the binary condition of occurrence versus non-occurrence of an activity.

Magnitude is the third primitive because an activity may occur and yet do so at an insufficient or excessive level. A production process may operate below required capacity; a release may exceed permitted limits; a transformation may deliver output at a rate that does not meet demand. Magnitude expresses the level, intensity, or capacity at which an activity is executed. It is distinct from quantity of forms because it pertains to the performance of the activity itself rather than to the count of resulting forms.

The identification of time, existence, and magnitude as primitive dimensions satisfies the criterion for primitive identification established in Section 3. These three primitives—time, existence, and magnitude—are necessary to describe all activity-level variation. Without time, the system cannot determine whether an activity occurs in the correct temporal relation to other activities. Without existence, the system cannot distinguish between activities that occur and those that do not. Without magnitude, the system cannot determine whether an activity operates at a level sufficient to produce the required outcome. Each dimension corresponds to a necessary condition for activity to fulfil its role in the system.

4.1 Continuous Variables at the Activity Level

Continuous scalar variables appear not only as properties of forms but also as outcomes of activities. Requirements frequently specify the level at which an activity must operate:

- a heating activity must reach a given temperature,
- a pump must achieve a specified pressure,
- a signal must be generated within a defined voltage range.

Such cases may suggest that scalar variables introduce a distinct dimension of activity-level deviation.

Proposition

Continuous variables at the activity level do not introduce a new deviation dimension. They are fully captured by magnitude.

Proof

An activity requirement specifies that an operation must occur at a certain level, within a defined range.

Failure of such a requirement implies that:

- the activity has occurred,
- but its level of operation does not satisfy the requirement.

The deviation therefore concerns the **intensity, capacity, or level of execution** of the activity.

This is precisely the definition of magnitude.

No additional dimension is required to express this failure.

Separation of Type and Metric

The distinction established in Section 3 applies equally at the activity level:

- deviation type identifies how an activity fails,
- deviation metric quantifies the extent of that failure.

Scalar variables determine the metric—degrees, bars, volts—but the type of deviation remains magnitude.

Consequence

The activity-level deviation basis remains:

$$\{E, T, M\}$$

complete and irreducible.

Continuous variables do not extend the grammar. They provide measurement scales within it.

Link to Form-Level Analysis

Section 3 established that scalar variables, when specified as properties of forms, reduce to structural deviation. The present section completes the argument: when specified as outputs of activities, they reduce to magnitude.

No scalar variable generates an independent deviation dimension at either level.

The sufficiency of the set follows from the observation that any deviation in the execution of an activity can be expressed along one of these three dimensions. If an activity occurs too early or too late, the deviation lies in time. If it does not occur, the deviation lies in existence. If it occurs at an insufficient or excessive level, the deviation lies in magnitude. No additional primitive is required to describe how an activity can fail in its execution.

The impossibility of a fourth primitive follows from reduction. Any proposed additional dimension must either introduce a new independent way in which an activity can fail or be reducible to one of the three existing dimensions. Consider common candidates. Sequence is reducible to time because it expresses ordering. Frequency is reducible to magnitude because it expresses rate. Duration is reducible to magnitude or time depending on whether it is treated as level or interval. Control or activation is reducible to existence because it determines whether the activity occurs. No candidate introduces a dimension that cannot be expressed in terms of time, existence, or magnitude. The set is therefore closed.

A crucial distinction must be maintained between activity and form. Time, existence, and magnitude describe how an activity is executed; they do not describe intrinsic properties of forms. A form that appears late does so because the activity that produced or delivered it was temporally misaligned. A form that is missing is the consequence of an activity that did not occur. A form that is insufficient in capacity reflects an activity that operated at inadequate magnitude. The ontology of activity therefore explains the origin of deviations that manifest at the boundary as properties of forms but do not belong to the form itself.

This distinction resolves a common ambiguity in problem solving. Without it, delays, absences, and capacity issues are often attributed directly to forms, leading to confusion between what a form is and how it was produced or transferred. By assigning temporal, existential, and magnitude deviations to activities, the science separates

intrinsic form defects from execution defects. This separation is essential for locating root causes and for constructing a coherent grammar of failure.

The ontology of activity complements the ontology of form. Together, they define the complete set of dimensions along which required and observed states can differ. Structure and quantity describe what the form is and how many instances exist. Time, existence, and magnitude describe how the activities that produce and handle those forms are executed. No additional dimensions are required.

The integration of these two ontologies yields a finite set of primitive deviation dimensions. Each deviation observed in a system must belong to one of these dimensions. The next section formalises this integration by deriving the complete grammar of deviation. From the five primitive dimensions—two of form and three of activity—the full set of deviation types is obtained. This result establishes that the ways in which failure can occur are not open-ended but bounded by the structure of form and activity themselves.

4.2 Proof of Activity Closure

Activity is defined as that which alters the state of a form. For such alteration to occur, four and only four conditions must be satisfied:

– constitution: there must exist a form whose state can be altered
– spatial relation: the form must be positioned relative to the system boundary
– temporal availability: the alteration must occur at a defined moment
– normative admissibility: the alteration must be permitted within the system

Each condition is necessary. Remove constitution and no object exists to act upon. Remove spatial relation and the system cannot distinguish internal from external states. Remove temporal availability and no change can be ordered or observed. Remove normative admissibility and the alteration is undefined within the system.

To establish exhaustiveness, consider any candidate fifth condition. Such a condition must either:

– describe a property of the form (reducing to constitution),
– describe its position (reducing to spatial relation),
– describe its occurrence (reducing to temporal availability), or
– describe its permissibility (reducing to normative admissibility).

No fifth independent condition exists.

The six activity classes arise from the distinct realisations of these four conditions across relational contexts.

Constitution governs transformation of identity, yielding **Transform**, the only activity that alters what a form is.

Spatial relation governs position relative to the system. A change of position within the system yields **Move**. A change across the system boundary yields two directional variants: **Acquire** (entry) and **Release** (exit).

Temporal availability governs persistence and moment-specific evaluation. Maintenance of a form across time yields **Store**. Evaluation of a form at a defined moment yields **Check**.

Normative admissibility constrains all activities but does not define a distinct transformation of state. It therefore governs the validity of activity without constituting an additional activity class.

These six activities—Store, Move, Acquire, Release, Transform, Check—exhaust all possible ways in which a form's state can be altered or maintained under the four necessary conditions. Any proposed seventh activity must either duplicate one of these relations or fail to satisfy the conditions. The activity set is therefore closed.

5. The Closure of Deviation

The preceding sections established two distinct ontologies: the ontology of form, defined by structure and quantity, and the ontology of activity, defined by time, existence, and magnitude. These five primitive dimensions—two attaching to forms and three to activities—define the complete set of ways in which a required state can differ from an observed state. The present section derives the full grammar of deviation from these primitives and establishes its closure.

A deviation is the difference between a required state and an observed state at the boundary of an activity. Since all states are expressed in terms of the five primitive dimensions, any deviation must correspond to a variation along one or more of these dimensions. The problem therefore reduces to enumerating the distinct modes in which each primitive can fail relative to a requirement.

5.1 Structural Deviations (S)

Structure captures what a form is. A structural deviation occurs when the identity or constitution of the observed form does not match the required form. It can be shown that structural deviation is not a single undifferentiated category but a finite family of distinct modes. These modes are irreducible to one another and collectively exhaustive.

Six structural deviations are required:

- **S1 — Wrong type**: the form is of an incorrect identity

- **S2 — Misaligned**: the form is correct in type but wrongly positioned or associated
- **S3 — Incomplete**: required elements are missing
- **S4 — Inconsistent**: internal contradictions exist within the form
- **S5 — Unstable**: the form does not persist in its required state
- **S6 — Defective**: the form lacks the capability to fulfil its function

Each of these represents a distinct failure mode of identity. No further structural deviation can be defined without collapsing into one of these six. A form cannot be wrong in any other fundamentally different way.

5.2 Quantitative Deviations (Q)

Quantity captures how many instances of a form are present relative to requirement. Only two primitive deviations exist:

- **Q− — Too little**
- **Q+ — Too much**

These two exhaust all possibilities of variation in multiplicity. Any apparent nuance—partial shortage, extreme excess—remains a variation in degree within one of these two polarities. No third quantitative deviation exists.

5.3 Temporal Deviations (T)

Time captures when an activity occurs relative to requirement. Two primitive deviations exist:

- **T− — Too early**
- **T+ — Too late**

These are mutually exclusive and exhaustive. Any temporal misalignment must occur in one direction or the other relative to the required time.

5.4 Existential Deviation (E)

Existence at the activity level captures whether the activity occurs at all. Only one primitive deviation is required:

- **E — Missing (non-occurrence)**

An activity either occurs or does not occur. There is no further subdivision at the primitive level. Apparent variations—partial execution, intermittent occurrence—reduce to combinations of existence with other primitives such as time or magnitude.

5.5 Magnitude Deviation (M)

Magnitude captures the level at which an activity operates. It expresses capacity, rate, or intensity. At the primitive level, magnitude deviation is expressed as:

- **M — Out of required range**

Unlike quantity and time, magnitude is not expressed in two separate symbolic forms because it is inherently bounded by a required range. Any deviation corresponds to operating below or above acceptable limits. Both cases are captured within the same primitive, since they are symmetric with respect to the requirement.

5.6 The Complete Deviation Set

The complete set of primitive deviations is therefore:

$$\Delta = \{E,\ S_1, S_2, S_3, S_4, S_5, S_6,\ M,\ Q^-, Q^+,\ T^-, T^+\}$$

with cardinality:

$$|\ \Delta\ | = 12$$

This set constitutes the full grammar of deviation.

5.7 Proof of Exhaustiveness

Each primitive dimension admits a finite set of deviation modes determined by how a requirement can fail. For structural deviation, the modes are not stipulated but derived from the failure of requirement primitives.

Structural requirements are defined through the primitives Reference, Assertion, and Condition. Each primitive admits distinct failure modes:

– Reference failure produces misidentification of the object or its context, yielding S1 — wrong type, and S2 — misaligned

– Assertion failure produces incorrect specification of what must be true, yielding S3 — incomplete, S4 — inconsistent, and S6 — defective

– Condition failure produces instability in the applicability of the requirement, yielding S5 — unstable

These six modes are necessary and sufficient because they exhaust the ways in which requirement primitives can fail. Removing any one eliminates a distinct class of structural error; adding another would require a new primitive or a subdivision of an existing failure mode, which would collapse into one of the six.

Other primitive dimensions admit fewer modes due to their structure:

– Quantity admits two modes because multiplicity can only deviate by deficit or excess
– Time admits two modes because ordering can only deviate by earliness or lateness
– Existence admits one mode because presence is binary
– Magnitude admits one mode because value can deviate from the required range

The complete deviation set is therefore:

$$\Delta = \{E,\ S_1,\ S_2,\ S_3,\ S_4,\ S_5,\ S_6,\ M,\ Q^-,\ Q^+,\ T^-,\ T^+\}$$
$$|\ \Delta\ | = 12$$

Suppose an additional deviation exists. It must correspond either to a new primitive dimension or to a new mode within an existing dimension. Any proposed additional deviation dimension must satisfy the primitive identification criterion established in Section 3, and no candidate does.

No new primitive dimension exists (Section 3), and no additional mode can be defined without reducing to one of the above. The deviation set is therefore exhaustive.

5.8 Proof of Irreducibility

Each deviation type is irreducible.

A structural deviation cannot be expressed as a quantity deviation because identity and multiplicity are independent. A temporal deviation cannot be expressed as magnitude because ordering is distinct from value. Existence cannot be decomposed into other primitives because absence is binary.

For any pair of deviation types $\delta_i, \delta_j \in \Delta$, neither can be expressed as a composition of the other without loss of information about the violated requirement.

The deviation grammar is therefore minimal and irreducible.

5.9 Separation of Form and Activity Deviations

A critical structural property of the grammar is the separation between form-level and activity-level deviations:

- **Form deviations**: S, Q
- **Activity deviations**: T, E, M

This separation preserves the ontology established in the previous sections. Structural and quantitative deviations describe intrinsic defects in forms. Temporal, existential, and magnitude deviations describe failures in the execution of activities. The two sets are not interchangeable. A delay cannot be reduced to a structural defect; a misclassification cannot be reduced to a timing issue.

5.10 Closure Result

The derivation of Δestablishes that the ways in which failure can occur are finite and completely specified. There is no residual category of deviation outside this set. Any observed failure must belong to one or more elements of Δ.

This result is decisive. It transforms deviation from an open-ended concept into a closed grammar. Once combined with the closed set of activities, it yields a complete enumeration of all possible failures. The next section performs this combination and establishes the full space of Negative Outcomes.

The science now possesses a finite language in which every failure can be expressed.

6. Closure of Failure — The 72 Negative Outcomes

The preceding section established that all deviations belong to a finite and closed grammar Δof twelve elements. Failure, however, is not defined by deviation alone. A deviation becomes a Negative Outcome only when it is observed at the boundary of a specific activity. The closure of failure therefore requires combining the grammar of deviation with the structure of activity.

Let A denote the set of activities. This set is finite and closed:

$$A = \{Store, Move, Acquire, Release, Transform, Check\}$$

Each activity defines a boundary at which forms are produced, received, transferred, or verified. It is at these boundaries that required and observed states are compared and deviation becomes observable. A Negative Outcome is therefore defined as the pairing of an activity with a deviation:

$$NO = A \times \Delta$$

Each element of this set represents a distinct way in which a required state can fail at a specific activity boundary.

6.1 Enumeration of the Failure Space

Since both $A$and Δare finite, their Cartesian product is finite. The cardinality of the Negative Outcome space is:

$$|\ NO\ | = |\ A\ | \times |\ \Delta\ | = 6 \times 12 = 72$$

These seventy-two elements constitute the complete space of observable failure.

Each element $(a, \delta) \in NO$specifies:

– the location of failure: the activity $a$at whose boundary the deviation is observed
– the nature of failure: the deviation δdescribing how the observed state differs from the required state

No additional dimensions are required. Every failure that can be observed in a system must correspond to one of these seventy-two positions.

6.2 Interpretation of the Negative Outcome Space

The significance of the result lies not in the number itself but in its closure. What appears in practice as an unbounded variety of failures reduces to a finite and structured set. Shortages, excesses, delays, missing outputs, incorrect transformations, and verification failures are not independent phenomena. They are instances of a single grammar applied at different activity boundaries.

For example:

– A shortage of parts at a storage location corresponds to $(Store \times Q^{-})$
– Excess inventory corresponds to $(Store \times Q^{+})$
– A delayed delivery corresponds to $(Move \times T^{+})$or $(Release \times T^{+})$, depending on the boundary
– A missing output corresponds to $(Transform \times E)$
– A defective product corresponds to $(Transform \times S)$
– A failed inspection corresponds to $(Check \times S)$

These examples illustrate that domain-specific language maps directly onto positions in the Negative Outcome space. The mapping is not heuristic; it is structural. Each observed failure occupies one and only one position in the space when correctly specified.

6.3 Uniqueness and Non-Ambiguity

A properly stated Negative Outcome is unique. Ambiguity arises only when failures are described at the wrong boundary or in composite terms. For instance, waiting time is not a primitive Negative Outcome; it is a consequence of one or more underlying deviations, typically temporal deviations at upstream activities. Similarly, rework or scrap are not primitive outcomes; they are responses to structural deviations in prior activities.

The requirement to express failure as an activity-bound deviation eliminates such ambiguity. Each Negative Outcome must be located at the boundary where the requirement is violated. Once correctly located, the deviation type follows from the grammar. The result is a one-to-one correspondence between observed failures and elements of NO.

6.4 Completeness of the Failure Space

The completeness of the Negative Outcome space follows from the closure of its components. Since all deviations are contained in Δand all activity boundaries are contained in A, any observable failure must correspond to an element of their product.

To suppose the existence of a failure outside this space would require either:

– a deviation not expressible in terms of the primitive dimensions, or
– an activity not included in the set A

The previous sections have shown that neither condition can occur. The primitive dimensions are exhaustive, and the activity set is complete. Therefore, the Negative Outcome space is closed.

6.5 Structural Consequence

The closure of failure has a direct consequence for the science. It establishes that failure is not an open-ended phenomenon requiring exploration but a finite set requiring enumeration. Every problem corresponds to a position in a known space. The task of problem solving is therefore not to invent possible failures but to correctly identify which element of the space is observed.

Such shift is fundamental. It replaces an exploratory approach with a classificatory one. The analyst no longer asks what could be wrong in an open sense but determines which of the seventy-two Negative Outcomes is present. Once identified, the search for causes proceeds within a finite and constrained domain.

6.6 Foundation for Causation

The closure of failure provides the necessary foundation for the derivation of causation. Causes must explain the occurrence of Negative Outcomes. If the set of Negative Outcomes is finite, the set of admissible causes must also be finite. The mapping between causes and outcomes can therefore be determined in advance.

The next section builds on this result by establishing the necessary conditions for causation. It shows that causes arise from failures in the supports that enable activities to produce required states, and that these failures themselves form a finite and structured space.

7. Structural Causation — Necessary Conditions

The closure of the Negative Outcome space establishes that all observable failures belong to a finite set of seventy-two activity-bound deviations. A science of causation must now explain how such outcomes arise. The question is no longer what kinds of failure can occur, but under what conditions they occur. The answer is structural: every

Negative Outcome arises from the failure of necessary conditions required for activities to produce and maintain forms in their required state.

An activity produces a required outcome only if it is supported. Support is not an incidental feature of execution; it is a structural requirement. No activity can produce a correct output in the absence of the means that define how it should operate, who should perform it, with what instruments, and on the basis of what information. These supports are not domain-specific artefacts but universal categories that exist in any organised system.

Four such supports are defined:

$$POTD = \{Process, Organisation, Tools, Data\}$$

Each element represents a necessary condition for the correct execution of an activity.

Process defines how the activity is to be performed. It specifies the sequence of steps, the transformations to be applied, and the logic governing execution. Without process, the activity has no defined method and cannot reliably produce the required form.

Organisation defines who performs the activity. It assigns responsibility, authority, and capability. Without organisation, the activity lacks an agent capable of executing the process.

Tools define with what means the activity is performed. They include physical equipment, systems, and any instrument required to carry out the process. Without tools, the activity cannot operate at the required level or precision.

Data defines on what basis the activity is performed. It includes specifications, parameters, instructions, and any informational input required to guide execution. Without data, the activity cannot determine the correct state to produce.

These four supports are jointly necessary. The absence or inadequacy of any one renders the activity incapable of producing the required outcome. Their completeness follows from the fact that any activity must have a defined method, an executing agent, means of execution, and information specifying the required state. No additional category of support is required, and none can be removed without making execution undefined.

The failure of support does not occur in arbitrary ways. Each element of POTD is itself an activity: a process is executed, organisation acts, tools operate, and data flows. As activities, their failure modes are governed by the universal activity deviation grammar derived in Sections 4 and 6. Three and only three primitive modes apply:

$$EMT = \{Existence, Magnitude, Timeliness\}$$

Existence refers to whether the activity occurs at all. A process may not be executed, a role may remain unassigned, a tool may not be operated, or required data may not be provided. In such cases, the supporting activity does not take place.

Magnitude refers to whether the activity operates at the required level. A process may be executed but at insufficient capacity or precision; a role may be filled but at inadequate scale of engagement; a tool may operate but below its required performance; data may be present but insufficient in scope or accuracy. In such cases, the activity occurs but produces a result outside the required range.

Timeliness refers to whether the activity occurs at the required time. A process may be executed too early or too late; a role may become available after the required moment; a tool may be deployed outside the required window; or data may arrive before or after it is needed. In such cases, the activity fails to satisfy its temporal requirement.

These three modes are exhaustive because they correspond exactly to the three primitive dimensions of activity deviation established in Section 4. Any failure of a supporting activity must correspond to non-occurrence, insufficient magnitude, or temporal misalignment. No fourth mode can be defined without reducing to one of these three.

The combination of supports and failure modes yields a finite set of structural failure conditions:

$$| POTD \times EMT | = 4 \times 3 = 12$$

These twelve positions represent all possible ways in which the necessary conditions for activity execution can fail.

7.1 Causation as Failure of Necessary Conditions

A Negative Outcome occurs when an activity fails to produce or maintain a required state. Such failure must arise from a deficiency in one or more of the supports that enable the activity. If all supports are present, correct, and timely, the activity has the means to produce the required outcome. A deviation therefore implies that at least one support has failed in one of the three modes.

Causation is thus defined structurally. A cause is not an arbitrary event or narrative explanation; it is a position within the space of support failures. Each cause corresponds to a specific combination of support type and failure mode.

For example:

-A missing instruction corresponds to $(Data \times Existence)$> -An under-specified procedure corresponds to $(Process \times Magnitude)$> - An unavailable operator

corresponds to $(Organisation \times Timeliness)$> - A machine operating below required capacity corresponds to $(Tools \times Magnitude)$

These examples illustrate that causes are not domain-specific descriptions but instances of a universal structure.

7.2 Closure of the Cause Space

The closure of the Negative Outcome space implies that the space of causes must also be finite. Since each Negative Outcome arises from a failure of necessary conditions, and since these conditions are fully described by the twelve combinations of $POTD \times EMT$, the total cause space is:

$$| Cause |=| NO |\times| POTD \times EMT |= 72 \times 12 = 864$$

These eight hundred and sixty-four positions constitute the complete set of structural causes at the operational level.

Each position represents a distinct way in which a particular Negative Outcome can arise from a specific failure of support. No additional causes exist outside this space. Any observed cause must correspond to one of these positions.

7.3 Necessity and Sufficiency

The causal framework satisfies both necessity and sufficiency.

It is necessary because no activity can produce a required outcome without all four supports functioning correctly. A deficiency in any support is sufficient to generate a deviation.

It is sufficient because all possible deficiencies are captured by the twelve combinations of support type and failure mode. No additional category of failure is required to explain any Negative Outcome.

7.4 Independence from Domain

The structure of causation is independent of domain. The same twelve support-failure combinations apply in manufacturing, healthcare, logistics, software, or any other system. What differs across domains is the instantiation of supports—specific processes, roles, tools, and data—not the structure of causation itself.

This independence is critical. It ensures that the science does not rely on domain knowledge to define the space of causes. Domain knowledge is required only to identify which specific artefact corresponds to a given structural position.

7.5 From Outcome to Cause

The closure of both the Negative Outcome space and the cause space establishes a direct relation between them. For any given Negative Outcome, the set of admissible causes is a subset of the twelve support-failure combinations. The next step is to determine which of these combinations can produce a given outcome.

Such mapping is not arbitrary. It is governed by the structure of activities and the way in which support failures propagate through them. The formalisation of this mapping will be developed in subsequent sections through the concept of governing forms and the Structural Reduction Matrix.

7.6 Structural Foundation

The result of this section is the establishment of causation as a closed and finite system. Causes are not discovered through exploration but identified within a predefined structure. Each cause corresponds to a specific failure of a necessary condition.

The science now possesses:

– a closed set of failures (72 Negative Outcomes)
– a closed set of causes (864 structural positions)

The remaining task is to connect these two sets through a formal mapping that determines which causes can produce which outcomes. This mapping will elevate the framework from enumeration to reasoning and complete the transition from description to science.

8. Closure of the Cause Space — The 864 Positions

The preceding section established that all causation arises from the failure of necessary conditions supporting activity. These conditions are defined by the four support types—Process, Organisation, Tools, and Data[5]—and the three modes of failure—Existence, Magnitude, and Timeliness. Their combination yields a finite set of twelve structural failure modes. The present section completes the derivation by combining these modes with the closed set of Negative Outcomes, thereby establishing the full space of operational causation.

A Negative Outcome is defined as an activity-bound deviation. Each Negative Outcome represents a distinct way in which a required state can fail at a specific activity boundary.

[5] Kepner & Tregoe (1965)

Each such outcome must be explained by a failure in one or more of the necessary supports. Since these supports can fail in twelve distinct ways, the total space of causal positions is given by the Cartesian product:

$$Cause = NO \times (POTD \times EMT)$$

with cardinality:

$$| Cause | = | NO | \times | POTD \times EMT | = 72 \times 12 = 864$$

These eight hundred and sixty-four positions constitute the complete space of operational causes.

8.1 Interpretation of the 864 Positions

Each position in this space represents a structurally distinct causal hypothesis. It specifies:

– a Negative Outcome: the observed failure at a given activity boundary
– a support type: which necessary condition has failed
– a failure mode: how that condition has failed

For example, consider a shortage observed at a storage activity, expressed as $(Store \times Q^{-})$. The twelve possible structural causes correspond to:

– Process absent, operating below required capacity, or mistimed
– Organisation missing, incapable, or unavailable
– Tools absent, operating below required performance, or deployed outside the required window
– Data absent, insufficient in scope or accuracy, or provided at the wrong time

Each of these represents a distinct position in the cause space. No additional causal category exists outside these twelve for that particular Negative Outcome.

This structure applies uniformly across all seventy-two Negative Outcomes. Each outcome has exactly twelve structurally distinct causal positions. The apparent diversity of real-world causes—supplier delays, incorrect forecasts, machine breakdowns, human errors—maps onto these positions. The diversity lies in the instantiation, not in the structure.

8.2 Completeness of the Cause Space

The completeness of the 864 positions follows directly from the closure of their components. Since:

– all failures are contained in the seventy-two Negative Outcomes, and
– all causes are contained in the twelve support-failure combinations

their product exhausts all possible causal explanations at the operational level.

To suppose the existence of a cause outside this space would require either:

– a failure of a support not included in $POTD$, or
– a mode of failure not reducible to Existence, Magnitude, or Timeliness

The previous section established that neither condition can occur. The set of supports is complete, and the set of failure modes is exhaustive. Therefore, the cause space is closed.

8.3 Necessity and Sufficiency of Positions

Each of the 864 positions represents a necessary form of causation in the sense that any actual cause must correspond to one of them. They are not all realised in a given system, but they are all admissible in principle.

They are also sufficient in the sense that no additional structural category is required to explain any Negative Outcome. A real-world cause may involve multiple positions—for example, both incorrect data and inadequate tools—but each component of that cause still belongs to the defined space.

The positions therefore form a complete basis for causal explanation. Complex causes are combinations of primitive positions, not additions to the space.

8.4 From Enumeration to Elimination

The significance of the 864 positions lies not in their enumeration but in their role in reasoning. Traditional problem solving begins with an open set of possible causes and attempts to identify plausible explanations. In contrast, the present framework begins with a closed set of possible causes. The task is not to generate hypotheses but to eliminate positions that are inconsistent with evidence.

For any given Negative Outcome, the initial candidate set consists of twelve positions. Evidence is used to determine which positions can be excluded. The process continues until one or more positions remain that account for the observed deviation.

Such transformation—from open-ended search to bounded elimination—is made possible by the closure of the cause space. Completeness is guaranteed by structure, not by exhaustive exploration.

8.5 Independence from Domain Content

The 864 positions are defined independently of domain content. They do not describe specific artefacts, roles, or processes. They describe structural relations between

failure and its necessary conditions. Domain knowledge is required only to map these positions to concrete instances.

For example, a forecast with insufficient scope is not a primitive cause; it is an instance of $(Data \times Magnitude)$. A machine operating below required output is not a primitive cause; it is an instance of $(Tools \times Magnitude)$. A machine that is not present at all is an instance of $(Tools \times Existence)$.

. The framework abstracts from domain language to reveal the underlying structure.

This abstraction is essential for universality. It ensures that the same causal framework applies across all systems, regardless of their specific content.

8.6 Limits of the Operational Cause Space

The 864 positions represent causation at the level of operational support. They describe how activities fail to produce required outcomes due to deficiencies in their enabling conditions. However, they do not yet locate the origin of those deficiencies. A missing process, an incorrect dataset, or an unavailable tool is itself the result of prior decisions, specifications, or governing actions.

The science therefore requires a further level of analysis. The next section introduces the governing-form ontology, which locates root cause at the level where requirements are defined and authorised. This extension does not replace the 864 positions but explains their origin. It connects operational failures to structural defects in the governing layer.

8.7 Structural Foundation for Further Derivation

The closure of the cause space completes the operational level of the science. At this stage, the framework possesses:

– a finite set of failures (72 Negative Outcomes)
– a finite set of causes (864 structural positions)

The remaining task is to establish how these causes arise and how they are related to one another within the system structure. This requires moving from operational supports to governing forms and from enumeration to structured mapping.

The next section performs this transition by defining the ontology of governing forms and locating root cause within it.

The derivation chain is summarised in Figure 2 in appendix A.

9. Governing-Form Ontology

The closure of the operational cause space establishes that all observable failures can be explained through a finite set of support deficiencies. Yet these deficiencies are not primitive in themselves. A missing process, an incorrect dataset, an unavailable tool, or an inadequately defined role does not arise spontaneously. Each reflects a prior failure in the way the system defines, authorises, and maintains its own conditions of operation. To complete the science, causation must therefore be extended beyond operational supports to the governing layer in which those supports are specified.

A governing form is any artefact that defines what must be true in the system. Governing forms do not execute work; they define the conditions under which work is to be executed. They include specifications, plans, requests, constraints, authorisations, and verification constructs. Each governs one or more aspects of activity by specifying required states for forms or required conditions for execution. Without governing forms, the system has no basis for determining correctness.

The governing layer is structured through a finite set of functions:

$$SFRCAV = \{Specification, Forecast/Plan, Request \\ /Trigger, Constraints, Authorisation, Verification\}$$

Each function corresponds to a distinct role in governing activity.

Specification defines what the required form must be. It establishes identity, composition, and acceptable states.

Forecast/Plan defines what is expected to occur over time. It establishes required quantities and timing.

Request/Trigger defines when an activity should be initiated. It activates execution based on need or condition.

Constraints define what is permitted or forbidden. They limit or shape possible actions.

Authorisation defines who is allowed to perform an action or approve a state. It establishes legitimacy.

Verification defines how correctness is assessed. It determines whether required states have been achieved.

These six functions are exhaustive. Any governing artefact in any system performs one or more of these roles. No additional governing function is required to define how systems specify, activate, constrain, authorise, or verify work.

Governing forms are themselves structured objects. Their internal logic is defined through requirement primitives[6]:

$$RACE = \{Reference, Assertion, Condition, Evidence\}$$

Each primitive corresponds to a necessary component of a requirement.

Reference identifies the form to which the requirement applies. Without reference, the requirement has no object.

Assertion specifies what must be true of the form. It defines the required state.

Condition specifies when the requirement applies. It determines applicability.

Evidence defines how compliance is established. It provides the basis for verification.

These primitives are necessary and sufficient. Any requirement that governs activity must identify its object, state what must be true, specify when it applies, and define how correctness is assessed. Remove any one of these elements, and the requirement becomes incapable of guiding or evaluating action.

Governing forms also exist within a lifecycle. They are created, activated, used, modified, and eventually terminated. The lifecycle states are:

$$LIFESPAN = \{\text{Legitimate,Inactive,Finished prematurely,Extended,}$$
$$\text{Substituted, Phantom, Abandoned, Neglected}\}$$

These states describe whether a governing form is valid, active, and appropriate within the system. A form may exist structurally but be illegitimate in its lifecycle—for example, an outdated specification still in use, or a required instruction never activated.

The governing-form universe is therefore defined as:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

Each element of this set represents a specific governing construct with a defined function, internal structure, and lifecycle state.

9.1 Root Cause at the Governing Level

The operational cause space identifies where support fails. The governing-form ontology identifies why those supports fail. A process is missing because no governing form specified it. Data is incorrect because the assertion within a specification is

[6] Zave & Jackson (1997)

wrong. A role is unavailable because authorisation or planning failed. Tools are inadequate because constraints or forecasts were improperly defined.

Root cause is therefore located at the governing level and defined as:

$$RC = T \times S \times GF$$

A root cause is a structural deviation in a governing form, produced through a transformation. The transformation corresponds to the activity that created or modified the governing form. The structural deviation corresponds to a defect in its requirement primitives. The governing form identifies the locus of that defect.

This definition distinguishes root cause from operational cause. Operational causes describe failures in execution. Root causes describe defects in the structures that define execution. Operational causes are therefore consequences of governing defects.

9.2 Structural and Lifecycle Dimensions

A governing form can fail in two fundamentally different ways. It may be structurally defective, meaning that its requirement primitives are incorrect or incomplete. Alternatively, it may be structurally correct but exist in an illegitimate lifecycle state.

These two dimensions are orthogonal. Structure describes what the governing form contains. Lifecycle describes whether it is valid and active. A specification may be perfectly defined yet obsolete. A request may be structurally incomplete yet active. Both cases produce failure, but through different mechanisms.

The science resolves this through a classification priority. When a governing form is in an illegitimate lifecycle state, its structural content is irrelevant because it does not participate correctly in the system. Lifecycle illegitimacy is therefore the dominant causal condition. When a governing form is legitimate but structurally defective, the structural defect is causal.

This distinction ensures that each causal position is uniquely classified, even when both types of defect are present.

9.3 Closure of the Governing-Form Space

The governing-form universe is finite because each of its components is finite. The six governing functions, the four requirement primitives, and the finite set of lifecycle states combine to form a closed space. Structural deviations apply to the requirement primitives, yielding a finite set of governing defects.

This closure extends the causal framework beyond the operational level. It shows that not only are failures and operational causes finite, but the structures that generate them are also finite.

9.4 From Governing Forms to Operational Failure

The role of governing forms in the science is to connect structural defects to observable outcomes. A defect in a governing form propagates through the system by influencing the supports of activities. An incorrect specification leads to incorrect data; an absent request prevents an activity from occurring; an invalid constraint limits execution improperly. These effects manifest at activity boundaries as Negative Outcomes.

The mapping from governing defects to observable failures is not arbitrary. It is determined by the structure of activities and the propagation of forms through them. This mapping will be formalised in subsequent sections through the Structural Reduction Matrix, which determines which governing defects can produce which Negative Outcomes.

9.5 Elevation of the Science and Closure of Causation

The introduction of governing forms elevates the science from a descriptive framework of failure and support deficiencies to a complete theory of causation. Operational causes identify failures in the supports of activity. These supports are not primitive: they are themselves defined by governing forms that specify how activities should be performed, by whom, under what conditions, and with what information.

The science therefore introduces a second level of structure in which these governing constructs are defined. Governing forms are fully specified by:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

This set is finite and closed. Any governing construct must belong to this space, and any defect in governance must correspond to a structural deviation within it.

The introduction of this level terminates causal regress. Operational causes point to deficiencies in supports; these supports are defined by governing forms; governing forms are fully specified within a closed ontology. No further level of causation is required. The regress from outcome to cause therefore terminates at the governing-form level. The science is complete in the sense that all causes are contained within a finite and closed structure.

At this stage, the science possesses three closed spaces:

– the space of failures:

$$| NO | = 72$$

– the space of operational causes:

$$| Cause | = 864$$

– the governing-form universe:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

The remaining task is to establish the mapping between these spaces, determining how governing defects give rise to operational causes and, ultimately, to observable failures.

A critical structural relation must therefore be made explicit. The operational cause space defined by $POTD \times EMT$ constitutes the projection of governing-form defects into execution. Each operational deficiency corresponds to a failure in a governing form whose structure is defined by $SFRCAV \times RACE \times LIFESPAN$. The operational space of 864 positions represents the manifestation of governing defects at the level of activity support, while the governing-form space provides the generative structure from which these manifestations arise.

The two are not alternative causal architectures but successive levels of the same system:

$$SFRCAV \times RACE \times LIFESPAN \longrightarrow POTD \times EMT$$

The governing-form structure defines root cause. The operational structure defines its expression.

The mapping expresses a relation between governing specification and operational manifestation. The elements of $POTD$ correspond to governing functions and primitives that define execution:

– Process corresponds to Specification and Forecast/Plan in their role of defining how activities are performed
– Data corresponds to the Reference and Assertion primitives that define what must be true
– Organisation corresponds to Authorisation, which defines who may act
– Tools correspond to the resource dimension of Specification and Constraints, which define available means

The failure modes EMT correspond to the universal activity deviation grammar: a supporting activity may fail to occur (Existence), may operate outside the required level (Magnitude), or may occur at the wrong time (Timeliness). Because each element of POTD is an activity, this is the correct and only grammar for its failure modes. The operational cause space is therefore the projection of governing-form defects into activity execution. The governing-form ontology defines the generative structure; $POTD \times EMT$ defines its operational expression.

10. The MNAD — System Structure

The preceding sections established three closed domains: the space of Negative Outcomes, the space of operational causes, and the governing-form universe. A final structural element is required to connect these domains within a system. Causes do not act in isolation; they propagate through the structure of activities and forms. A governing defect produces an operational deficiency only if its effects can reach the activity boundary at which a deviation is observed. The science therefore requires a formal representation of system structure that makes such propagation explicit.

The MNAD provides this representation. The MNAD is not a behavioural model or a simulation of execution. It is a structural description of how forms move through activities under the influence of governing forms. It defines the pathways along which deviations can propagate and thereby determines which causal positions are admissible for a given Negative Outcome.

10.1 Definition of the MNAD

A system is composed of activities connected by the flow of forms. Each activity produces, receives, or transforms forms, and these forms pass from one activity to another. The MNAD represents this structure as a directed network:

$$G_{MNAD} = (A, E)$$

where:

– $A$is the set of activity nodes
– $E \subseteq A \times A$is the set of directed edges representing form transfers between activities

Each edge corresponds to the movement of a form from one activity to another. The MNAD therefore captures the topology of the system: which activities are connected and how forms flow between them.

This representation is deliberately minimal. It does not encode the internal logic of activities or the content of forms. It encodes only the structural relationships necessary for propagation.

10.2 Forms as Carriers of State

Forms carry the state produced by activities. When a form leaves an activity, it embodies the result of that activity's execution. When it enters a subsequent activity, it becomes the input to that activity. The state of forms therefore propagates through the network defined by G_{MNAD}.

A deviation introduced at one point in the system can influence downstream activities if the affected form is transmitted along the corresponding edges. Conversely, if no path exists between two activities, a deviation at one cannot affect the other.

The MNAD thus defines the channels through which causation operates. It determines not whether a deviation exists, but whether it can reach a given boundary.

10.3 Connectivity and Reachability

A governing defect or operational deficiency is relevant to a Negative Outcome only if it can influence the activity at which the deviation is observed. This condition is formalised as connectivity.

An element x(whether a governing form or an operational support) is said to be connected to an activity $a$if there exists a directed path in $G_{MNAD}$from the activity or artefact influenced by $x$to a.

Connectivity defines reachability. If no path exists, the effect of $x$cannot propagate to a, and $x$cannot be a cause of the Negative Outcome observed at a.

This condition is purely structural. It does not depend on the magnitude of the effect or the likelihood of propagation. It depends only on the existence of a path.

10.4 Admissibility Condition

The MNAD introduces a necessary condition for causation:

A structural defect is admissible as a cause of a Negative Outcome only if it is connected to the activity at which the deviation is observed.

This condition eliminates a large portion of the causal space for any given outcome. While the total space of operational causes contains 864 positions, only those that are structurally connected through $G_{MNAD}$to the relevant activity can be considered.

Connectivity therefore acts as a structural filter. It removes causally isolated positions without reference to evidence or probability. The elimination is mechanical and determined entirely by system structure.

10.5 Independence from Domain Content

The MNAD abstracts from domain-specific detail. It does not represent the physical layout of a factory, the organisational chart of a company, or the architecture of a software system. It represents only the logical connections between activities through the flow of forms.

This abstraction is essential for universality. The same principles of connectivity apply regardless of whether the forms are physical goods, information records, or authorisations. What matters is not the nature of the form but the existence of a path through which its state can propagate.

10.6 Role of Governing Forms in the MNAD

Governing forms influence the MNAD indirectly by shaping how activities are executed and how forms move between them. A specification determines what form should be produced; a plan determines when and in what quantity; a request triggers execution; constraints limit possible actions; authorisations permit or deny transitions; verification validates outcomes.

A defect in a governing form alters the behaviour of one or more activities. This alteration affects the forms produced or transferred, which then propagate through the MNAD. The MNAD thus provides the medium through which governing defects are translated into observable deviations.

10.7 Structural vs Behavioural Models

The MNAD must be distinguished from behavioural or simulation models. It does not predict how often an event will occur or how a system will behave under varying conditions. It does not incorporate probabilities, delays, or dynamic interactions. Its role is strictly structural.

The MNAD answers a single question: can an effect propagate from one point in the system to another? If the answer is no, the causal relation is impossible. If the answer is yes, the relation is admissible and must be considered.

This distinction reinforces the epistemic position of the science. Causation is determined by structure, not by probability or frequency. The MNAD defines the space of possibility, not the likelihood of occurrence.

10.8 Foundation for Mapping

The MNAD completes the structural prerequisites for mapping causes to outcomes. The science now possesses:

– a closed set of Negative Outcomes
– a closed set of causal positions
– a structural representation of system connectivity

The remaining step is to combine these elements into a formal mapping that determines, for each governing defect, which Negative Outcomes it can produce. This mapping must account for both the nature of the deviation and the reachability conditions imposed by the MNAD.

The next section introduces this mapping in the form of the Structural Reduction Matrix.

11. The Structural Reduction Matrix (SRM)

The preceding sections established the complete domains required for a science of causation: a finite set of Negative Outcomes, a finite set of causal positions, and a structural representation of system connectivity through the MNAD. The remaining task is to determine how these domains relate. Not every causal position can produce every Negative Outcome. A governing defect may be connected to an activity and yet be incapable of producing the specific form of deviation observed. The science therefore requires a formal mechanism that determines admissibility: which causes can, in principle, produce which outcomes.

This mechanism is the Structural Reduction Matrix (SRM).

11.1 Definition of the SRM

The Structural Reduction Matrix is defined as:

$$SRM: GF \times NO \rightarrow \{0,1\} \times R$$

where R is the set of route signatures representing admissible propagation paths within the MNAD.

A value of 1 indicates that the governing defect $g \in GF$ is structurally admissible as a cause of the Negative Outcome $no \in NO$. A value of 0 indicates that the relation is structurally impossible.

The route signature encodes the structural pathway through which the effect propagates across activities. Admissibility alone determines whether a relation is possible; the route signature determines how that relation is realised within the system structure.

The SRM therefore encodes possibility, not probability. It does not measure likelihood, frequency, or severity. It determines whether a causal relation can exist at all.

11.2 The Need for Reduction

The governing-form universe is finite but large. For any given Negative Outcome, many governing defects exist in principle. Without reduction, the set of candidate causes would remain too broad for effective reasoning.

The SRM reduces this set by eliminating all governing defects that cannot produce the observed deviation. This reduction is not heuristic. It does not rely on experience, judgement, or statistical inference. It is determined entirely by the structure of the deviation and the connectivity of the system.

For each pair (g, no), admissibility is determined in advance. The SRM is therefore a pre-computed structure: a complete mapping between the governing-form universe and the Negative Outcome space.

11.3 Two Necessary Conditions

The SRM encodes two necessary conditions for causation:

– Projection (identity perturbation)
– Connectivity (reachability in the MNAD)

A governing defect is admissible as a cause of a Negative Outcome if and only if both conditions are satisfied.

Projection

Projection concerns whether a governing defect can produce the type of deviation observed.

Each Negative Outcome corresponds to a deviation in one of the primitive dimensions: structure, quantity, time, existence, or magnitude. A governing defect must be capable of perturbing the identity terms associated with that deviation.

For example:

– A structural deviation requires that the governing form affects specification or transformation of identity
– A quantitative deviation requires that the governing form affects planning, allocation, or thresholds of quantity
– A temporal deviation requires that the governing form affects scheduling or triggering
– An existential deviation requires that the governing form affects activation or presence of activity
– A magnitude deviation requires that the governing form affects capacity or level of execution

If a governing defect does not act on the relevant dimension, it cannot produce the corresponding deviation.

Projection therefore eliminates all governing defects that cannot perturb the identity terms of the Negative Outcome.

Connectivity

Connectivity concerns whether the effect of the governing defect can reach the activity at which the deviation is observed.

Let $G_{MNAD} = (A, E)$be the system structure. A governing defect $g$is connected to an activity $a$if there exists a directed path in $G_{MNAD}$from the activity or artefact influenced by $g$to a.

If no such path exists, the defect cannot influence the outcome at that activity boundary.

Connectivity therefore eliminates all governing defects that are structurally isolated from the relevant activity.

11.4 Combined Admissibility

The SRM implements the condition:

$$SRM(g, no) = 1 \iff Projection(g, no) \wedge Connectivity(g, no)$$

Both conditions are necessary.

Projection without connectivity yields a defect that could produce the correct type of deviation but cannot reach the relevant activity.

Connectivity without projection yields a defect that can reach the activity but cannot produce the required type of deviation.

Only their conjunction yields an admissible causal relation.

11.5 Reduction of the Causal Space

For a given Negative Outcome no, the set of admissible governing defects is:

$$GF_{adm}(no) = \{g \in GF \mid SRM(g, no) = 1\}$$

This set is a strict subset of the governing-form universe.

The SRM reduces the space of possible causes by eliminating all structurally impossible relations. The reduction is deterministic. Given the definition of $no$and the structure of G_{MNAD}, the admissible set is uniquely determined. No judgement or interpretation is required to perform this reduction.

Completeness follows because every governing defect capable of producing a deviation must act on the identity terms of that deviation, and such action is captured by the projection condition. Correctness follows because no defect can influence an activity without a path in the MNAD, and such influence is captured by connectivity. The SRM therefore includes all admissible relations and excludes all impossible ones.

11.6 Independence from Evidence

The SRM operates independently of evidence. It does not confirm or refute causes; it determines which causes are possible. Evidence is applied only after reduction, to distinguish among admissible causes.

This separation defines two stages of reasoning:

– Structural elimination: removal of all impossible causes
– Empirical confirmation: identification of realised causes within the admissible set

The first stage is governed entirely by the SRM. The second stage operates within a finite and pre-defined set.

11.7 Pre-Computation and Universality

The SRM is not constructed anew for each problem. It is derived from the structure of the science and applies universally. For each combination of governing function, requirement primitive, lifecycle state, and deviation type, admissibility can be determined once and for all.

The SRM can therefore be treated as a fixed matrix. It encodes the complete relation between governing defects and Negative Outcomes across all systems. Domain-specific information enters only through:

– the mapping of governing forms to concrete artefacts
– the structure of the MNAD

11.8 From Enumeration to Mapping

The introduction of the SRM completes the transition from enumeration to structured reasoning.

The science now possesses:

– a finite set of failures
– a finite set of causes
– a structural representation of system connectivity
– a mapping that determines admissibility

Causation is no longer a matter of listing possible explanations. It becomes a traversal of a defined structure. For any Negative Outcome, the admissible causes are known in advance. The role of analysis is to determine which of these causes are realised.

11.9 Foundation for Reasoning Operations

The SRM provides the foundation for all reasoning operations within the science:

– Diagnosis: $NO \rightarrow GF$
– Prediction: $GF \rightarrow NO$
– Stress testing: $GF_{sys} \rightarrow NO_{sys}$
– Preventive design: modification of GF to reduce admissible NO

Each operation is a traversal of the same structure in a different direction.

The next section formalises these operations and shows how the same framework supports all forms of causal reasoning.

The complete architecture is illustrated in Figure 1 of Appendix A.

12. The Causal Reasoning System

The Structural Reduction Matrix establishes the admissible relation between governing defects and observable failures. Combined with the closure of the Negative Outcome space, the cause space, and the system structure represented by the MNAD, it completes the formal architecture of the science. What remains is to define how this architecture is used. The present section introduces the four reasoning operations that constitute the complete causal system: diagnosis, prediction, stress testing, and preventive design.

These operations are not independent methods. They are different traversals of the same structure. No new ontology is introduced, no additional assumptions are required, and no separate reasoning mechanism is added. Each operation uses the same elements—Negative Outcomes, governing forms, the MNAD, and the SRM—and differs only in the direction of traversal.

12.1 Diagnosis — From Outcome to Cause

Diagnosis begins with an observed Negative Outcome. The objective is to determine which governing defects can explain the deviation.

Let a Negative Outcome be defined as:

$$no = (a, \delta)$$

The SRM provides the admissible set of governing forms:

$$GF_{adm}(no) = \{g \in GF \mid SRM(g, no) = 1\}$$

This set is finite and determined a priori. Diagnosis proceeds by evaluating each element of this set against evidence. Elements that are inconsistent with observed conditions are eliminated. The process continues until one or more governing defects remain that account for the deviation.

Diagnosis therefore consists of two stages:

– structural reduction (performed by the SRM)
– empirical elimination (performed through evidence)

No hypothesis generation is required. The space of possible causes is known in advance. Completeness is guaranteed because all admissible causes are included in $GF_{adm}(no)$.

12.2 Prediction — From Cause to Outcome

Prediction reverses the direction of reasoning. It begins with a known or suspected governing defect and determines which Negative Outcomes can arise from it.

For a governing form $g \in GF$, define the predictive set:

$$NO_{adm}(g) = \{no \in NO \mid SRM(g, no) = 1\}$$

This set represents all outcomes that are structurally possible given the defect in g. Prediction does not determine which outcome will occur or when it will occur. It determines what can occur.

The symmetry between diagnosis and prediction follows directly from the definition of the SRM. The same mapping is traversed in the opposite direction. No additional mechanism is introduced.

12.3 Stress Testing — From System to All Outcomes

Diagnosis and prediction operate on individual elements: a single Negative Outcome or a single governing defect. Stress testing extends the reasoning to the system level.

Let $GF_{sys} \subseteq GF$ be the set of governing forms instantiated in the system. The set of admissible outcomes is:

$$NO_{sys} = \bigcup_{g \in GF_{sys}} NO_{adm}(g)$$

This set represents all Negative Outcomes that can occur given the current governing structure. It includes both observed and unobserved failures. Stress testing therefore reveals latent exposure: outcomes that have not yet occurred but are structurally possible.

No assumption about probability or frequency is required. The result is purely structural.

12.4 Preventive Design — Modification of Structure

Preventive design operates on the governing structure itself. Its objective is to modify the system so that certain Negative Outcomes are no longer admissible.

Let the system be defined by its governing forms GF_{sys}. A modification produces a new set GF'_{sys}. The corresponding outcome set becomes:

$$NO'_{sys} = \bigcup_{g \in GF'_{sys}} NO_{adm}(g)$$

Preventive design consists in transforming the governing structure such that:

$$NO'_{sys} \subset NO_{sys}$$

That is, the set of admissible outcomes is reduced.

The key principle is that prevention does not act on outcomes directly. It acts on the governing forms that make those outcomes possible. By correcting structural defects, enriching requirement primitives, or restoring lifecycle legitimacy, the system removes the conditions under which certain deviations can occur.

12.5 Unity of the Reasoning System

The four operations—diagnosis, prediction, stress testing, and preventive design—form a complete system. They are not separate techniques but different views of the same structure:

– Diagnosis: $NO \rightarrow GF$
– Prediction: $GF \rightarrow NO$
– Stress testing: $GF_{sys} \rightarrow NO_{sys}$
– Preventive design: $GF_{sys} \rightarrow GF'_{sys} \rightarrow NO'_{sys}$

Each operation uses the SRM and the MNAD. Each operates within the same closed domains. The differences lie only in the direction and scope of traversal.

12.6 Structural Completeness

The reasoning system is complete because it covers all possible relations between causes and outcomes:

– any observed outcome can be traced to its admissible causes
– any governing defect can be mapped to its admissible consequences[7]

[7] Pearl (2009)

– any system can be evaluated for all outcomes it admits
– any modification can be assessed for its impact on those outcomes

No additional operation is required to complete the system. All reasoning about failure and causation reduces to these four traversals.

12.7 From Science to Application

The introduction of the reasoning system marks the transition from structure to use. The preceding sections established the ontology and mappings of the science. The present section shows how those elements are applied without introducing new concepts.

The science is therefore not a collection of methods but a unified system. Once the structure is defined, all reasoning follows from it. Diagnosis, prediction, stress testing, and prevention are not separate practices but expressions of the same underlying relations.

The next section examines the implications of this system for diagnosis in particular, focusing on determinism, completeness, and the elimination of hypothesis-driven reasoning.

12.8 Worked Example — Store × Q^{-}

Consider the Negative Outcome:

$$no = (Store, Q^{-})$$

This represents a shortage at the storage boundary.

The SRM yields a finite admissible set of governing defects affecting quantity specification, replenishment triggering, and constraint definition. These defects map to governing functions within:

– Forecast/Plan (quantity definition)
– Request/Trigger (replenishment activation)
– Constraints (allocation and limitation rules)

Instantiation within the MNAD identifies concrete artefacts such as demand forecasts, reorder policies, and allocation rules.

Diagnosis proceeds by elimination. For each admissible governing position, the analyst verifies:

– whether the governing form exists
– whether its assertion is correct
– whether it is applied at the correct time
– whether its lifecycle is legitimate

For example, an underestimated forecast corresponds to a defect in the Assertion primitive of a Forecast/Plan governing form. A missing reorder trigger corresponds to a defect in the Request/Trigger function. An allocation constraint blocking replenishment corresponds to a defect in Constraints.

The evaluation continues until the confirmed governing defects collectively explain the observed shortage.

The result demonstrates that multiple governing defects may contribute to the same quantitative deviation. Empirically, dominance is often observed in planning and constraint structures, particularly where financial or policy constraints override replenishment logic.

This example shows that the abstract machinery yields concrete and non-trivial causal distributions. The framework does not merely classify failure; it produces structured explanations grounded in the governing architecture of the system.

For the Negative Outcome $no = (Store, Q^{-})$, the SRM yields an admissible set including:

$$GF_{adm}(no) = \{g_1, g_2, g_3, \dots\}$$

where:

– g_1: Forecast assertion deficit (quantity underestimated)
– g_2: Missing replenishment trigger
– g_3: Constraint blocking supply

Evidence is applied sequentially. Suppose inventory records show that reorder triggers were correctly issued. Then:

$$g_2 \notin GF_{conf}$$

Suppose allocation rules reveal a constraint limiting replenishment. Then:

$$g_3 \in GF_{conf}$$

Suppose forecast data shows systematic underestimation. Then:

$$g_1 \in GF_{conf}$$

The confirmed set is:

$$GF_{conf} = \{g_1, g_3\}$$

Causal integrity is satisfied when:

$$\sum_{g \in GF_{conf}} c(g) = D$$

Assume the observed shortage is:

$$D = 100$$

Forecast deficit contributes:

$$c(g_1) = 60$$

Constraint limitation contributes:

$$c(g_3) = 40$$

Then:

$$60 + 40 = 100 = D$$

Causal integrity is satisfied, and diagnosis terminates under Condition A.

The shortage is therefore explained by the combined effect of planning deficiency and constraint limitation. The example demonstrates elimination, confirmation, and closure.

13. Deterministic Diagnosis and Closure

The causal reasoning system establishes that diagnosis is a traversal from an observed Negative Outcome to a finite set of admissible governing defects. The present section formalises the properties of this traversal. It establishes that diagnosis, when performed under the presuppositions of the science, is deterministic and complete. Reasoning is not the construction of explanations but the elimination of impossibilities within a closed space.

13.1 From Open Search to Structural Elimination

Traditional approaches to diagnosis begin from an undefined set of possible causes and attempt to identify plausible explanations. The process is inherently open-ended. Hypotheses are generated, tested, revised, and replaced. Completeness cannot be demonstrated because the space of possibilities is not bounded.

In the present framework, the situation is reversed. For any given Negative Outcome no, the Structural Reduction Matrix defines the admissible set:

$$GF_{adm}(no) = \{g \in GF \mid SRM(g, no) = 1\}$$

This set is finite and known in advance. Diagnosis therefore does not begin with hypothesis generation. It begins with a complete set of structurally admissible causes. The task is to determine which elements of this set are consistent with observed conditions.

Diagnosis proceeds by elimination. Each candidate $g \in GF_{adm}(no)$is evaluated against evidence. If the required governing form does not exist or exists in a state inconsistent with the candidate position, the candidate is eliminated. The process continues until the remaining elements account for the observed deviation.

The shift is fundamental. Reasoning is no longer exploratory but reductive. The analyst does not search for causes; the analyst removes those that are impossible.

13.2 Causal Integrity

A Negative Outcome corresponds to a measurable deviation:

$$D = R - O$$

This deviation may be produced by one or more governing defects. Each confirmed defect contributes to the deviation. Let $c(g)$denote the contribution of a governing defect g. The principle of causal integrity requires that:

$$\sum_{g \in GF_{conf}} c(g) = D$$

where $GF_{conf}$is the set of confirmed causes.

Diagnosis achieves closure when the total contribution of confirmed causes equals the observed deviation. At this point, the deviation is fully explained. No further causes are required.

This condition follows directly from the definition of deviation as a measurable gap. A complete explanation must account for the entire gap.

13.3 Exhaustion of the Admissible Set

A second termination condition arises from the finiteness of the admissible set. Diagnosis may reach a point at which all elements of $GF_{adm}(no)$have been evaluated. If all candidates are either confirmed or excluded, no further structural causes remain to be considered.

This condition establishes that diagnosis cannot continue indefinitely. The process must terminate because the set of candidates is finite.

13.4 Termination Conditions

Diagnosis admits two termination conditions:

– **Condition A — Causal Integrity**

$$\sum_{g \in GF_{conf}} c(g) = D$$

– **Condition B — Exhaustion**

$GF_{adm}(no)$ fully evaluated

Under Condition A, diagnosis is complete. The deviation is fully explained by the confirmed governing defects.

Under Condition B, diagnosis terminates structurally: no further admissible causes remain. If the deviation is not fully explained, the issue lies not in causal reasoning but in the presuppositions of the analysis.

13.5 Presupposition Conditions

The determinism of diagnosis depends on two presupposition conditions:

– correctness of the Negative Outcome
– completeness of the MNAD and governing-form instantiation

A Negative Outcome is correctly specified if:

– the activity boundary corresponds to the point at which the deviation is observed
– the deviation type corresponds to the measurable gap between required and observed states
– the statement reflects the operational boundary, not a downstream symptom

The MNAD is complete if all relevant activities, connections, and governing forms are represented such that every admissible governing pattern has at least one artefact instantiation.

If these conditions are not satisfied, the admissible set may be incomplete or incorrectly defined.

13.6 Scope of Judgement

Within the presupposition conditions, diagnosis requires no judgement. The admissible set is determined structurally, and elimination is governed by evidence. No selection among possible causes is performed by the analyst.

Judgement remains only at the level of presupposition:

– determining whether the Negative Outcome is correctly specified
– determining whether the system representation is complete

This distinction is essential. The science eliminates first-order judgement (selection among causes) but does not eliminate second-order judgement concerning the correctness of inputs.

13.7 Determinism of Diagnosis

Under correct presuppositions, diagnosis is deterministic. Given a Negative Outcome $no$and a complete system representation:

– the admissible set $GF_{adm}(no)$is uniquely determined
– the elimination process yields a unique set of confirmed causes

Different analysts operating on the same inputs will arrive at the same result. The outcome does not depend on experience, intuition, or interpretation.

13.8 Falsifiability

The determinism of diagnosis is falsifiable. The framework would be invalidated if:

– a correctly specified Negative Outcome exists
– the system representation is complete
– all admissible causes are evaluated
– the deviation remains unexplained

Such a case would imply that either the failure space or the governing-form space is incomplete.

The absence of such cases follows from the closure established in earlier sections. Since all failures and causes are contained within finite sets, and their relation is defined by the SRM, the framework is complete by construction.

This diagnostic falsification condition is a specific instance of the broader falsifiability criteria defined in Section 14.5, where failure of closure or admissibility would invalidate the framework.

13.9 Consequence for Problem Solving

The implications are immediate. Problem solving ceases to be an open-ended search for explanations and becomes a structured process of elimination. Completeness is guaranteed by the closure of the causal space. The analyst's role shifts from generating possibilities to verifying which of the predefined possibilities are realised.

This transformation replaces heuristic reasoning with deterministic analysis and ensures that no admissible cause is overlooked.

13.10 Transition

The determinism of diagnosis completes the operational aspect of the science. The remaining sections situate the framework within a broader scientific context, examine its relation to existing approaches, and articulate its implications for practice.

14. Scientific Status of Causonomy

The preceding sections established a complete system: a finite ontology of failure, a finite space of causes, a structural representation of systems, and a mapping that determines admissibility. The present section establishes that this system satisfies the criteria of a science. The objective is not to assert novelty but to demonstrate that the framework possesses the defining properties that distinguish a science from a method.

14.1 Definition of a Science

A science is characterised by three elements:

– a defined object[8]
– a finite or bounded domain
– necessary relations governing that domain

A further requirement is falsifiability: the capacity for the framework to be invalidated by counterexample.

Methods, by contrast, provide procedures or heuristics for addressing problems. They do not define a closed object or guarantee completeness. They operate within open domains and rely on judgement or experience to guide application.

[8] Popper (1959)

The question is therefore whether the framework developed in this paper satisfies these criteria.

14.2 Defined Object

The object of the science is the Negative Outcome: the deviation between a required and an observed state at the boundary of an activity.

Let $R(f)$ denote the required state of a form f, and $O(f)$the observed state. Deviation is defined as:

$$D(f) = R(f) - O(f)$$

A Negative Outcome exists if and only if:

$$D(f) \neq 0$$

A Negative Outcome is formally expressed as:

$$NO = (a, \delta), a \in A,\ \delta \in \Delta$$

The object is independent of domain content. It applies equally to physical, informational, and organisational systems.

The object is observable and measurable. Failure is not described in narrative or interpretive terms but as a relation between two defined states.

The existence of a defined object satisfies the first condition of a science.

14.3 Finite Domain

The domain of the science is finite at multiple levels:

– the deviation set:

$$|\ \Delta\ | = 12$$

– the activity set:

$$|\ A\ | = 6$$

– the Negative Outcome space:

$$|\ NO\ | = |\ A\ | \times |\ \Delta\ | = 72$$

– the operational cause space:

$$| Cause | = 72 \times 12 = 864$$

– the governing-form universe:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

The governing-form universe has cardinality:

$$| GF | = 6 \times 4 \times 8 \times 6 = 1{,}152$$

corresponding to SFRCAV × RACE × LIFESPAN × structural deviation modes.

Combining six governing functions, fourteen requirement primitives, and eight lifecycle states yields a governing-form universe of 1,152 structurally distinct root-cause positions.

Each of these sets is derived from necessary conditions rather than empirical enumeration. The finiteness of the domain ensures that the space of possible failures and causes is bounded. No additional elements exist outside these sets.

The presence of a finite domain satisfies the second condition of a science.

14.4 Necessary Relations

The relations within the domain are defined by necessity rather than by observation or correlation.

– The grammar of deviation follows from the primitive dimensions of form and activity
– The Negative Outcome space follows from the Cartesian product:

$$NO = A \times \Delta$$

– The operational cause space follows from:

$$Cause = NO \times (POTD \times EMT)$$

– The admissibility mapping is defined by:

$$SRM(g, no) = 1 \iff Projection(g, no) \wedge Connectivity(g, no)$$

These relations do not depend on empirical frequency or statistical inference. They are derived from the structure of the system. Given the definitions, the relations must hold.

The presence of necessary relations satisfies the third condition of a science.

14.5 Falsifiability

The framework is falsifiable. It can be invalidated if any of the following conditions are observed:

– a failure that cannot be expressed within the deviation set Δ
– a cause that does not correspond to a failure within $POTD \times EMT$ or the governing-form space
– a governing defect $g \in GF$ that produces an outcome no such that $SRM(g, no) = 0$
– a Negative Outcome no for which:

$$GF_{adm}(no) = \emptyset$$

These conditions provide precise criteria for refutation. The framework does not rely on interpretive flexibility; it makes structural claims that can be tested.

14.6 Independence from Domain

A defining property of a science is independence from specific instances. The framework applies across domains because it is defined at the level of structure.

The same:

– activity set A
– deviation grammar Δ
– governing-form structure GF
– admissibility mapping SRM

apply regardless of whether the system concerns manufacturing, healthcare, logistics, or information processing.

Domain knowledge is required only to instantiate forms and governing artefacts. It is not required to define the structure itself.

14.7 Comparison with Established Sciences

The structure of the framework aligns with the development of other sciences.

In geometry, properties are derived from a finite set of axioms. In information theory[9], communication is described through a bounded measure. In each case, the domain is closed and governed by necessary relations.

The present framework performs an analogous operation for failure and causation. It defines a finite space and derives the relations within it.

The similarity lies not in subject matter but in method: the transition from empirical description to structural necessity.

14.8 Distinction from Methods

Existing approaches to problem solving—root cause analysis[10], fault trees, probabilistic models—operate as methods. They provide procedures for analysing failures but do not define a closed domain.

They rely on:

– enumeration
– expert judgement
– statistical inference

Completeness cannot be guaranteed because the space of possible causes is not bounded.

The present framework differs in kind. It does not generate explanations; it defines the complete set of admissible explanations:

$$GF_{adm}(no)$$

Reasoning proceeds by elimination within this set.

The framework therefore replaces methods rather than extending them.

14.9 Consequences of Scientific Status

The establishment of problem solving as a science yields the following properties:

– **Completeness**: all failures and causes are contained within finite sets
– **Determinism**: reasoning yields identical results given identical inputs
– **Universality**: the framework applies across domains
– **Predictive capacity**:

$$NO_{adm}(g)$$

[9] Shannon (1948)
[10] Rooney & Vanden Heuvel (2004)

can be derived for any governing defect

– **Preventive capability**:

$$NO'_{sys} \subset NO_{sys}$$

can be achieved through structural modification

These properties follow from structure, not from method.

14.10 Conclusion of Scientific Status

The framework satisfies all criteria of a science. It defines a precise object, operates within a finite domain, establishes necessary relations, and admits falsification.

Problem solving is therefore no longer a practice guided by methods but a domain governed by structure.

The remaining sections examine the practical implications of this transformation and position the framework relative to existing approaches.

15. Relation to Existing Approaches

The comparison that follows is not methodological but ontological. The objective is not to compare tools or procedures but to compare the structure of the domains on which those tools operate. The present framework defines a closed and finite domain of failure and causation. Existing approaches operate within open and unbounded domains.

15.1 Comparative Structure of Problem-Solving Frameworks

Legend

✓ = present and explicit
~ = present implicitly or heuristically
✗ = absent
Open = unbounded / context-dependent
Closed = finite and provably complete

Table 1 — Comparative Structure

Framework	Activity Set	Deviation Set	Negative Outcome (Relational)	Output Failure Space	Support Type	Structural Failure Mode	Root Cause	Causal Link (Root Cause → NO)
New Science of Problem Solving	SMARTC (Closed)	12 deviations (Closed)	✓ (Activity × Deviation)	Closed (72 NO)	POTD (Closed)	EMT (Closed)	(T \times S \times GF) *(1,152 — governing level)*	Formal, necessary-condition based (Closed)

Framework	Activity Set	Deviation Set	Negative Outcome (Relational)	Output Failure Space	Support Type	Structural Failure Mode	Root Cause	Causal Link (Root Cause → NO)
HAZOP[11]	~ Process nodes (Open)	~ Guide words (Open)	✗	Open	✗	✗	✗	✗
STPA[12]	~ Control actions (model-dependent)	~ UCAs (Open)	✗	Open	~ Causal factors	✗	✗	✗ (scenario-based)
FMEA[13]	~ Functions / steps	✗	✗	Open	~ Contributing factors	~ Failure modes	✗	✗
Fishbone / Ishikawa[14]	✗	✗	✗	Open	~ Categories	✗	✗	✗
Five Whys[15]	✗	✗	✗	Open	✗	✗	~ Narrative	✗
Accident Investigation	✗	✗	✗	Open	~ Factors	✗	✗	✗
Statistical / Six Sigma	✗	✗	✗	Open	✗	✗	✗	✗
Systems Thinking	✗	✗	✗	Open	~ Interactions	✗	✗	✗
Expert Systems[16]	✗	✗	✗	Open (case-based)	~ Rules	✗	✗	✗
Foundational Ontologies	✗	✗	✗	Open	~ Norm violation	✗	✗	✗

The root cause count of 1,152 refers to the governing-form level defined by:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

combined with structural deviation $S_1 \dots S_6$.

The operational cause space defined earlier contains:

$$| Cause | = 864 = 72 \times 12$$

These are not competing counts but distinct levels of the same structure: governing origin and operational manifestation.

[11] IEC 61882 (2016)
[12] STPA (Leveson, 2011)
[13] IEC 60812 (2018)
[14] Ishikawa (1986)
[15] Ohno (1988)
[16] Feigenbaum (1977)

15.2 Families of Problem-Solving Approaches[17] and Their Structural Limits

Table 2 — Structural Limits

Tool Family	Representative Methods	What They Do Well	Structural Assumption	Structural Limitation	Why They Cannot Terminate Diagnosis
New Science of Problem Solving	Ontology + elimination	Guarantees completeness	Failure is finite and necessary	None at diagnosis level	Terminates by exhaustion
Idea Generation	Brainstorming	Generates ideas	Causes must be imagined	No admissibility rule	Space always open
Categorisation	Fishbone	Organises thinking	Causes fit buckets	Buckets are heuristic	Missing causes undetectable
Risk Enumeration	FMEA, HAZOP	Lists risks	Past approximates future	Enumeration incomplete	Omitted causes invisible
Failure Mechanisms	Engineering taxonomies	Describes breakdown	Damage explains cause	Confuses NO with cause	No causal constraint
Forensic Reconstruction	Accident reports	Reconstructs events	Chains explain origin	Chains extend indefinitely	No stopping rule
Statistical Analysis	SPC, Six Sigma	Detects patterns	Correlation implies cause	Correlation ≠ necessity	Absence unobservable
Systems Thinking	STPA	Explains interactions	Interaction explains failure	No boundary on interactions	Never exhaustive
Expert Systems	Rule-based[18]	Encodes knowledge	Past cases sufficient	Novel cases absent	Silence ≠ exclusion
Human Factors	HFACS	Identifies conditions	Responsibility diffuses	Conditions proliferate	No admissibility
Ontologies	DOLCE, BFO	Classify reality	Being can be categorised	No failure ontology	No causal elimination

15.3 Methods Based on Enumeration

Enumeration-based approaches[19] generate lists of possible causes. Their defining characteristics are:

[17] Ackoff (1979) *(systems thinking baseline framing)*
[18] Russell & Norvig (2010)
[19] Stamatis (2003)

– the space of causes is not defined in advance
– completeness cannot be demonstrated

The causal space remains open. New causes can always be proposed, and no rule exists to determine whether the set is complete.

The present framework replaces enumeration with identification within a finite set:

$$GF_{adm}(no)$$

Completeness is guaranteed structurally.

15.4 Methods Based on Probability

Probabilistic approaches[20] model frequency, not structure.

They:

– estimate likelihood
– rely on historical data

They do not define what can occur.

The present framework defines:

$$NO_{adm}(g)$$

independently of probability.

Probability operates only after structural possibility is established.

15.5 Fault Trees and Logical Decomposition

Logical methods decompose failure into event structures but rely on analyst-defined nodes[21].

Completeness depends on:

– identification of all events
– correctness of decomposition

The present framework fixes both primitives and relations in advance through:

$$SRM(g, no)$$

The analyst does not construct the structure.

[20] Pearl (2009)
[21] Vesely et al. (1981)

15.6 System-Theoretic Approaches

System approaches[22] analyse interactions and control structures.

They:

– model dynamics
– identify unsafe interactions

They do not define a finite failure ontology.

The present framework differs by:

– fixing the failure space
– constraining interactions structurally

The MNAD represents interaction without expanding the ontology.

15.7 Data-Driven Methods

Data-driven approaches[23] identify patterns in observed behaviour.

They:

– detect realised deviations
– analyse historical traces

They cannot enumerate unobserved failures.

The present framework defines the complete failure space before observation:

$$NO = A \times \Delta$$

Data is used only for confirmation.

15.8 Summary of Structural Difference

The distinction can be expressed along three axes:

– **Domain**:
Open (methods) vs Closed (science)

– **Reasoning**:
Generation vs Elimination

– **Completeness**:
Unknown vs Guaranteed

These differences are structural, not incremental.

---

[22] Leveson (2011)
[23] van der Aalst (2016)

15.9 Complementarity

Existing approaches remain useful, but their role changes:

– statistical methods operate on a defined space
– data methods provide evidence
– system models instantiate the MNAD
– engineering methods implement solutions

The framework defines the space within which these operate.

15.10 Consequence for Practice

Problem solving shifts from exploration to classification.

Instead of asking:

“What could be wrong?”

the analyst determines:

$$no \in NO$$

and then:

$$GF_{adm}(no)$$

This removes dependence on recall and hypothesis generation.

15.11 Transition

The framework has now been positioned relative to existing approaches. The remaining section synthesises the results and restates the central claim: that failure and causation form a closed domain governed by necessary relations.

16. Practical Implications and Applications

The preceding sections established a closed and necessary structure for failure and causation and clarified its distinction from existing approaches. The present section examines the consequences of this structure for practice. The objective is not to introduce new methods but to show how existing activities—diagnosis, prediction, system evaluation, and design—are transformed when performed within a closed causal framework.

16.1 Reframing Diagnosis

Diagnosis is no longer an open-ended search but a process of elimination within a finite set. The analyst begins by identifying the Negative Outcome and retrieving the corresponding admissible set:

$$GF_{adm}(no)$$

This set is complete by construction. Each element corresponds to a specific governing defect that must be evaluated against evidence.

Diagnosis proceeds through verification. For each $g \in GF_{adm}(no)$, the analyst determines whether the corresponding governing form exists and whether its structure and lifecycle state are consistent with the defect represented by g. Candidates inconsistent with observed conditions are eliminated.

The consequences are immediate:

– completeness is guaranteed because all admissible causes are considered
– efficiency is increased because structurally impossible causes are excluded in advance

The elimination of hypothesis generation reduces variability between analysts. Diagnosis becomes reproducible and independent of individual experience.

16.2 Directed Evidence Collection

The closure of the causal space transforms the role of evidence. Evidence is no longer used to suggest possible causes but to confirm or eliminate predefined ones. Each governing defect corresponds to specific observable conditions.

For example:

– a missing specification corresponds to absence of required data
– an incorrect plan corresponds to mismatch between required and planned quantities
– an inactive request corresponds to absence of triggering conditions
– an illegitimate lifecycle state corresponds to the presence of obsolete or inactive governing forms

Evidence collection becomes directed. Rather than gathering data broadly, the analyst seeks specific confirmations or contradictions for each admissible position.

This direction reduces the volume of data required and increases the precision of analysis. The structure determines what must be observed; the analyst verifies whether it is present.

16.3 Structural Prediction

Prediction is defined as the forward traversal from governing defects to Negative Outcomes. Given a governing defect g, the set:

$$NO_{adm}(g) = \{no \in NO \mid SRM(g, no) = 1\}$$

defines all outcomes that are structurally possible.

In practice, this allows analysts to evaluate the consequences of identified deficiencies before they manifest. A known defect in planning, constraint definition, authorisation, or specification can be analysed to determine which deviations may arise.

Prediction concerns possibility rather than likelihood. It defines the space of outcomes that the system admits. This enables proactive identification of structural risk embedded in governing forms.

16.4 System Exposure Analysis

Stress testing extends prediction to the system level. Let $GF_{sys} \subseteq GF$ denote the set of governing forms instantiated in the system. The corresponding set of admissible outcomes is:

$$NO_{sys} = \bigcup_{g \in GF_{sys}} NO_{adm}(g)$$

This set defines the system's exposure.

Exposure includes outcomes that have not yet occurred. A system may operate without incident while still admitting certain failures. Stress testing reveals these latent possibilities.

In practice, this enables evaluation of system robustness. The analyst can determine which Negative Outcomes are structurally possible and assess whether they are acceptable. Exposure becomes a property of system design rather than a function of observed performance.

16.5 Preventive Design

Preventive design operates by modifying governing forms to alter the set of admissible outcomes. The objective is not to reduce the frequency of failure but to remove the structural conditions under which certain failures can occur.

Let GF_{sys} be the current governing structure. A modification produces a new structure GF'_{sys}, and therefore a new outcome set:

$$NO'_{sys} = \bigcup_{g \in GF'_{sys}} NO_{adm}(g)$$

Prevention is achieved when:

$$NO'_{sys} \subset NO_{sys}$$

The effect is structural. Certain Negative Outcomes become impossible because the governing conditions that allowed them have been removed.

In practice, preventive design involves:

– correcting structural defects in governing forms
– completing requirement primitives (RACE)
– restoring lifecycle legitimacy
– introducing missing governing constructs

Prevention operates on admissibility, not on outcomes themselves.

16.6 Reduction of Dependence on Expertise

A central implication of the framework is the reduction of dependence on domain expertise[24] in the initial stages of problem solving. Identification of the Negative Outcome and retrieval of admissible causes require only correct classification within the defined ontology.

Expertise remains necessary for:

– interpreting domain-specific evidence
– understanding the content of governing forms
– implementing corrective actions

However, the structure of reasoning is independent of expertise. This separation increases accessibility and consistency.

16.7 Standardisation of Analysis

The closed structure enables standardisation. Since the spaces of failure and causation are fixed, and the mapping between them is defined, diagnostic and predictive processes can be standardised across systems.

[24] Deming (1986)

Standardisation does not imply uniformity of solutions but uniformity of reasoning. Different systems instantiate different governing forms and produce different outcomes, but the structure of analysis remains constant.

This property supports scalability. The same framework can be applied across organisations and domains without modification.

16.8 Integration with Data and Systems

The framework can be integrated with existing systems and data sources. The MNAD can be derived from process representations, and governing forms can be mapped to artefacts in enterprise systems. Data can be used to confirm or refute specific causal positions.

Such integration does not alter the structure of the science. It provides a means of operationalising it. Data enhances verification but does not define the causal space.

16.9 Transformation of Practice

The cumulative effect of these implications is a transformation of practice:

– from exploration to classification
– from hypothesis generation to elimination
– from reactive analysis to structural anticipation
– from domain-dependent reasoning to domain-independent reasoning

Problem solving becomes a process governed by structure rather than by method.

16.10 Transition

The practical implications demonstrate that the framework is not only theoretically complete but operationally applicable. The final section synthesises the results and restates the central claim: that failure and causation form a closed domain governed by necessary relations, and that this domain constitutes a science.

17. Conclusion

The objective of this paper has been to establish a science of failure and causation grounded in structural necessity rather than empirical enumeration or methodological practice. The argument proceeded by defining a precise object, deriving its primitive dimensions, establishing closure at each level, and constructing the relations that govern the domain. The result is a finite and complete framework within which all failures and their causes can be expressed.

The point of departure was the definition of failure as a Negative Outcome: the measurable deviation between a required state $R(f)$and an observed state $O(f)$at the boundary of an activity, such that:

$$D(f) = R(f) - O(f), D(f) \neq 0$$

and formalised as:

$$NO = (a, \delta)$$

This definition removes ambiguity by locating failure in a measurable relation rather than in descriptive language. From this object, the primitive ontology of form and activity was derived. Forms vary in structure and quantity; activities vary in time, existence, and magnitude. These five dimensions are necessary and sufficient. Their combination yields a closed grammar of twelve deviations.

The integration of this grammar with the closed set of activities produces a finite space of seventy-two Negative Outcomes:

$$| NO | = 72$$

Every observable failure corresponds to one element of this space. No additional category of failure exists outside it. The apparent diversity of real-world problems is thus reduced to a finite and structured set.

Causation was then derived as the failure of necessary conditions supporting activity. The four support types---Process, Organisation, Tools, and Data---each being a governed activity, fail according to the universal activity deviation grammar: Existence, Magnitude, and Timeliness. Their combination yields twelve structural failure positions. Their combination with the seventy-two Negative Outcomes produces a closed space of eight hundred and sixty-four operational causes:

$$| Cause | = 864$$

Each represents a distinct way in which a failure can arise.

The analysis was extended to the governing layer, where root cause is located. Governing forms define the conditions under which activities operate. Their structure is defined by governing functions, requirement primitives, and lifecycle states:

$$GF = SFRCAV \times RACE \times LIFESPAN$$

The governing-form universe is finite, yielding 1,152 structurally distinct root-cause positions. Structural defects in governing forms, produced through transformation, constitute root causes. Operational failures are thus traced to defects in the structures that define system behaviour.

The MNAD provides the structural representation of systems, defining the pathways through which forms and their states propagate. The Structural Reduction Matrix establishes the mapping between governing defects and Negative Outcomes:

$$SRM(g, no) = 1 \iff Projection(g, no) \wedge Connectivity(g, no)$$

This mapping determines admissibility and reduces the causal space for any given outcome to a finite set.

From these elements, a complete reasoning system was derived. Diagnosis, prediction, stress testing, and preventive design are not distinct methods but different traversals of the same structure. Diagnosis proceeds from outcome to cause through elimination within a finite set. Prediction proceeds from cause to outcome by identifying admissible consequences. Stress testing evaluates the full set of outcomes admitted by a system's governing structure. Preventive design modifies that structure to eliminate admissible failures.

The framework satisfies the criteria of a science. It defines a precise object, operates within a finite domain, establishes necessary relations, and admits falsification. It is independent of domain content and applies universally across systems. Its completeness ensures that no admissible failure or cause lies outside its defined domain.

The practical consequence is a transformation of problem solving. The open-ended search for causes is replaced by classification within a closed space. Reasoning becomes deterministic under correct presuppositions. Prediction becomes structural rather than probabilistic. Prevention operates by altering governing conditions rather than by reacting to observed failures.

The significance of this result lies in the unification of explanation, prediction, and design within a single framework. Failure is no longer treated as an empirical phenomenon to be explored but as a structured domain to be understood. Causation is no longer inferred but derived. The science provides not only a means of analysing problems but a foundation for designing systems in which certain problems cannot occur.

The framework is falsifiable: it would be invalidated by any failure, cause, or admissible relation that lies outside the defined spaces or violates the structural mapping. This condition establishes its scientific status.

The development presented here constitutes the initial formalisation of this science. Further work may extend its application, refine its representations, and integrate it with existing systems and data. The central claim is complete: failure and causation form a closed domain governed by necessary relations, and that domain admits a scientific treatment.

Appendix A — Structural Diagrams

Figure 1 — Complete Architecture of the Causal System

(see Section 11)

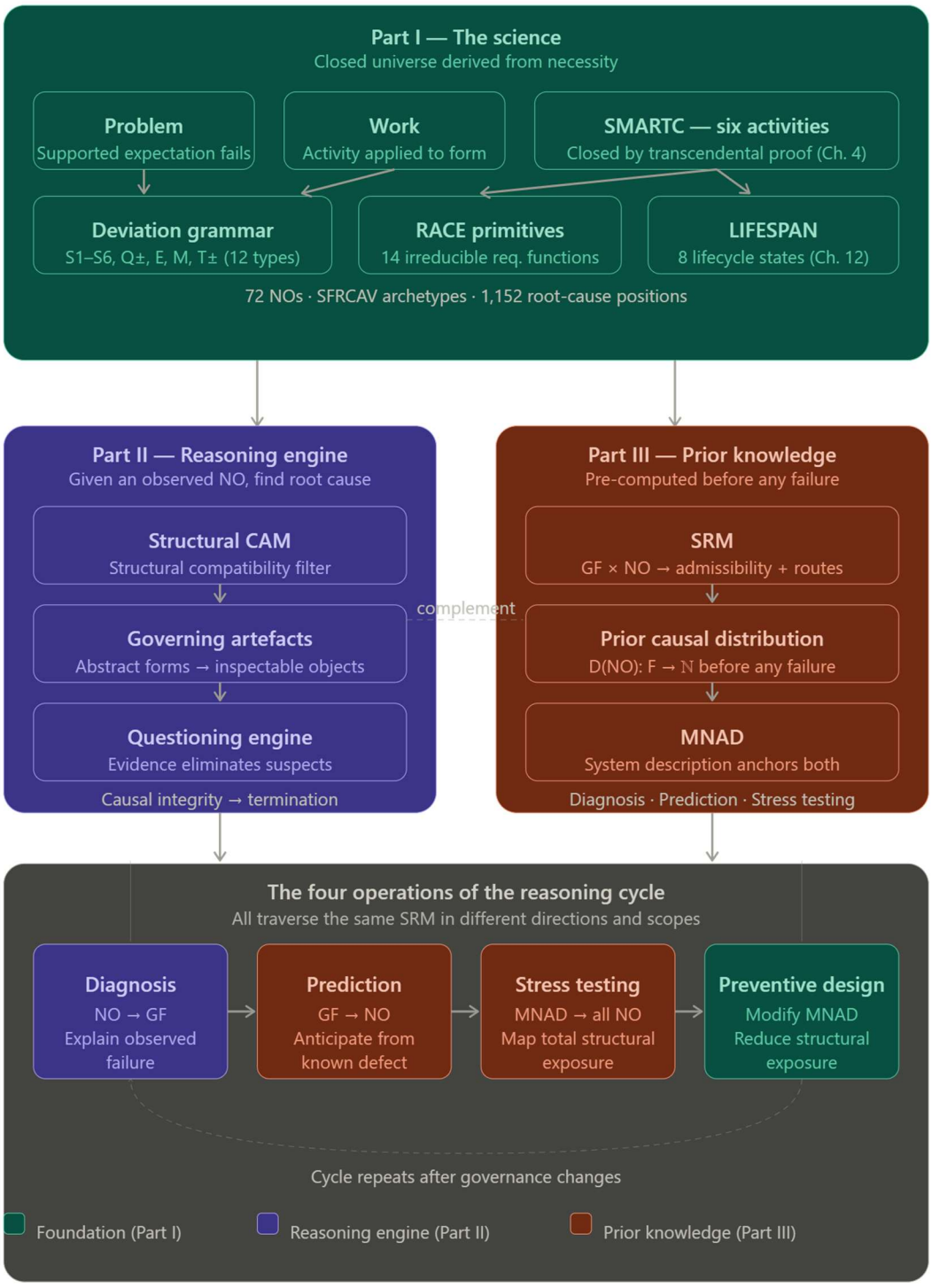
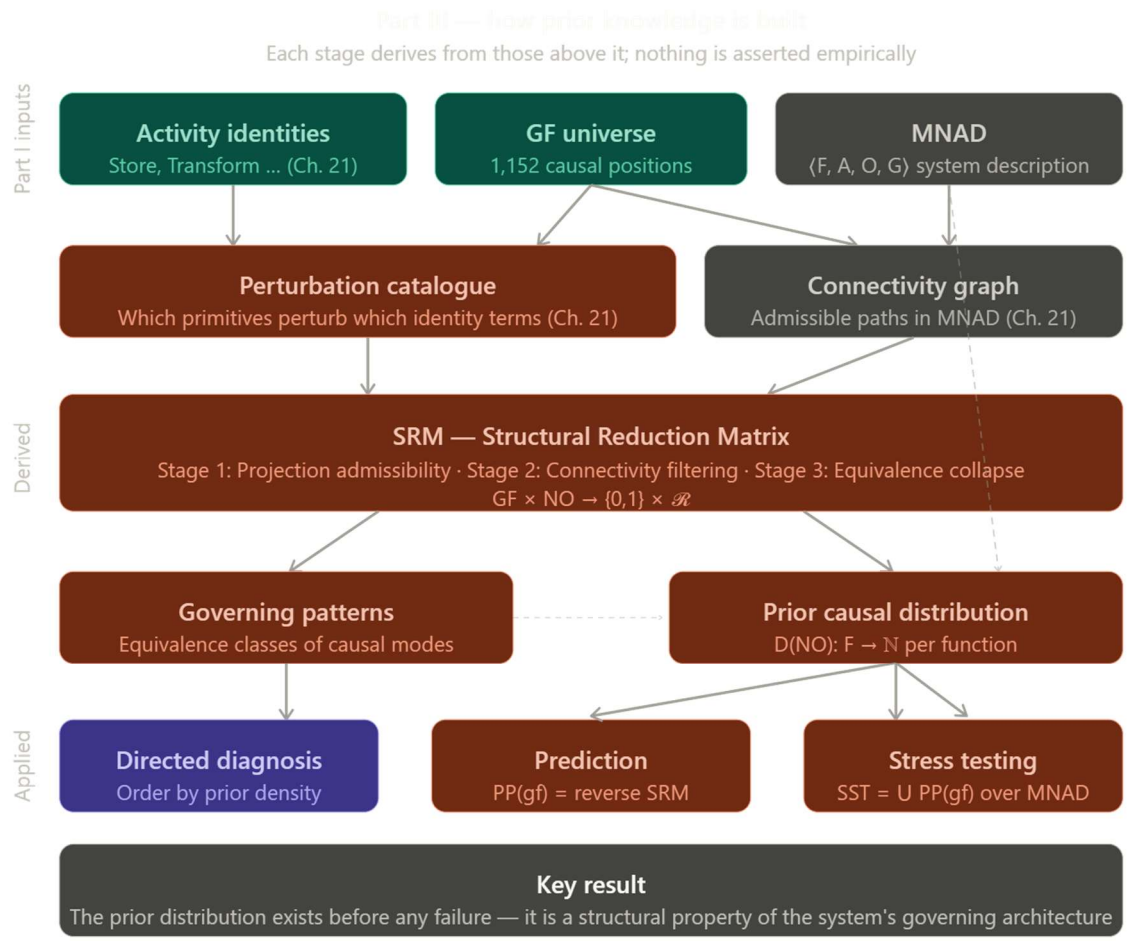


Figure 2 — Derivation Chain from Ontology to Causation

(see Sections 3–8)



References

1. Immanuel Kant, *Critique of Pure Reason*, 1781/1998
2. Claude Shannon, “A Mathematical Theory of Communication,” 1948
3. Karl Popper, *The Logic of Scientific Discovery*, 1959
4. Immanuel Kant, *Critique of Pure Reason*, 1781/1998
5. Aldo Gangemi et al., “Sweetening Ontologies with DOLCE,” 2002
6. Kepner & Tregoe, *The Rational Manager*, 1965
7. Zave & Jackson, “Four Dark Corners of Requirements Engineering,” 1997
8. Judea Pearl, *Causality*, 2009
9. Karl Popper, *The Logic of Scientific Discovery*, 1959
10. Karl Popper, *The Logic of Scientific Discovery*, 1959
11. Claude Shannon, “A Mathematical Theory of Communication,” 1948
12. Nancy Leveson, *Engineering a Safer World*, 2011
13. International Electrotechnical Commission, IEC 60812, 2018
14. International Electrotechnical Commission, IEC 61882, 2016
15. Ishikawa, *Guide to Quality Control*, 1986
16. Ohno, *Toyota Production System*, 1988
17. Vesely et al., *Fault Tree Handbook*, 1981
18. van der Aalst, *Process Mining*, 2016
19. Deming, *Out of the Crisis*, 1986
20. Ackoff, “The Future of Operational Research,” 1979
21. Stamatis, *Failure Mode and Effect Analysis*, 2003
22. Rooney & Vanden Heuvel, “Root Cause Analysis,” 2004
23. Judea Pearl, *Causality*, 2009
24. Judea Pearl, *Causality*, 2009
25. Vesely et al., *Fault Tree Handbook*, 1981
26. Nancy Leveson, *Engineering a Safer World*, 2011
27. Bertalanffy, *General System Theory*, 1968

28. Hastie, Tibshirani & Friedman, *The Elements of Statistical Learning*, 2009

29. van der Aalst, *Process Mining*, 2016

30. Provost & Fawcett, *Data Science for Business*, 2013

31. Womack & Jones, *Lean Thinking*, 1996

32. Ohno, *Toyota Production System*, 1988

33. Feigenbaum, “The Art of Artificial Intelligence,” 1977

34. Russell & Norvig, *Artificial Intelligence: A Modern Approach*, 2010

35. Deming, *Out of the Crisis*, 1986