

# The AEGON Algebra

## A Finite Semantic Algebra of Failure Classes for Deterministic System Interpretation

Adrian Diamond  
AAD Systems  
[adrian@aadsystems.com](mailto:adrian@aadsystems.com)

Version 1.0  
Written January 19, 2026  
<https://papers.aadsystems.com/aegon-algebra-v1>

### Abstract

Modern distributed systems exhibit unbounded scale and complexity, yet their observable behaviors remain structurally constrained. This paper introduces the AEGON algebra, a finite semantic algebra over system failure modes, demonstrating that system behavior admits a total deterministic classification independent of system size. We formalize a finite ontology of failure classes, define a partial order over these classes, and show how policy actions can be compiled from semantic classifications. The AEGON algebra provides a foundation for deterministic system interpretation, policy generation, and operational reasoning without reliance on probabilistic or heuristic models.

## Contents

<b>1 Structural Generators of the AEGON Algebra</b>	<b>3</b>
<b>2 Classification as Functorial Interpretation</b>	<b>3</b>
<b>3 Finite Semantic Closure of System Behavior</b>	<b>3</b>
<b>4 The Failure-Class Lattice and Thin Category Structure</b>	<b>4</b>
<b>5 Policy Compilation Semantics</b>	<b>4</b>
<b>6 Formal Semantic Examples</b>	<b>4</b>
6.1 Example 6.1: Single-Regime Classification and Compilation . . . . .	4
6.2 Example 6.2: Multi-Regime Join and Canonical Reconciliation . . . . .	5
6.3 Example 6.3: Explicit Neutral Outcome (No-Action as a First-Class Result) . . . . .	5
<b>7 Minimal AEGON Policy DSL and JSON Bundle Target</b>	<b>5</b>
7.1 Five-Construct Minimal Policy DSL (Sketch) . . . . .	5
7.2 DSL Example Compiling to JSON Policy Bundle . . . . .	5
<b>8 Interpretive Remarks on Classification Totality and Semantic Neutrality</b>	<b>6</b>



# 1 Structural Generators of the AEGON Algebra

**Definition 1.1 (Failure Class).** A failure class is a semantic atom representing a violated structural invariant of a system.

**Definition 1.2 (Failure-Class Ontology).** Let  $F = \{F_0, F_1, \dots, F_{13}\}$  be a finite, closed ontology of failure classes, where  $F_0$  denotes the neutral (no-failure) class.

**Definition 1.3 (Observation Space).** Let  $O$  denote the space of *normalized* system observations (e.g., structured signals, derived invariants, or canonical payloads), where normalization enforces representation invariance with respect to irrelevant operational variation. Elements of  $O$  are assumed to be sufficient for semantic classification.

**Remark 1.1.** This paper covers (i) the AEGON cloud application (deterministic classifier), and (ii) the AEGON compiler/language (policy compilation). These are distinct artifacts unified by the same semantic algebra.

## 2 Classification as Functorial Interpretation

**Definition 2.1 (Deterministic Classifier).** A deterministic classifier is a total function

$$C : O \rightarrow F$$

such that for identical  $o \in O$ , repeated evaluation yields identical output in  $F$ .

**Definition 2.2 (Failure-Class Category).** Define a category  $\mathcal{F}$  whose objects are the failure classes in  $F$  and whose morphisms represent semantic escalation relations (defined formally in §4).

**Definition 2.3 (Functorial View of Classification).** When  $O$  is regarded as a category  $\mathcal{O}$  of observations (with structure-preserving maps between observations), classification can be viewed as a functor

$$C : \mathcal{O} \rightarrow \mathcal{F}.$$

**Corollary 2.1 (Repeatability).** For all  $o \in O$ ,  $C(o)$  is stable under re-evaluation:  $C(o) = C(o)$  under all evaluation contexts.

*Proof.* Determinism is a definitional property of  $C$ .

## 3 Finite Semantic Closure of System Behavior

**Theorem 3.1 (Finite Semantic Closure).** System behavior (as represented in  $O$ ) admits a total deterministic classification into the finite ontology  $F$ , independent of system size.

**Lemma 3.2 (Semantic Coverage Principle).** If the failure-class ontology  $F$  is finite and closed with respect to admissible observations in  $O$ , then the classifier  $C : O \rightarrow F$  induces a complete semantic interpretation of system state for the purposes of policy generation.

*Proof.* By totality of  $C$  and closure of  $F$ , every admissible observation maps to exactly one policy-relevant semantic regime.

*Proof.* Since  $C : O \rightarrow F$  is total and  $F$  is finite, every observation maps to exactly one class in a finite codomain.

**Corollary 3.1 (Bounded Interpretive Complexity).** Even when operational state space is unbounded, interpretive state space is bounded by  $|F|$ .

*Proof.* The classifier's image is a subset of  $F$ .

## 4 The Failure-Class Lattice and Thin Category Structure

**Definition 4.1 (Failure-Class Partial Order).** Let  $\leq$  be a partial order on  $F$  such that  $F_i \leq F_j$  means class  $F_j$  semantically dominates or escalates  $F_i$ .

**Semantic Interpretation.** The partial order  $\leq$  does not encode temporal ordering, causality, or probabilistic likelihood. Rather,  $F_i \leq F_j$  expresses that the semantic regime  $F_j$  strictly dominates or subsumes  $F_i$  with respect to system interpretation and policy relevance.

**Definition 4.2 (Join-Semilattice (Policy-Relevant)).** Assume  $(F, \leq)$  admits binary joins  $\vee$  (least upper bounds) whenever policy composition must reconcile multiple triggered regimes.

**Lemma 4.1 (Thinness of the Failure-Class Category).** The category  $\mathcal{F}$  induced by  $(F, \leq)$  is thin: for any  $F_i, F_j \in F$  there exists at most one morphism  $F_i \rightarrow F_j$ .

*Proof.* In a poset-category,  $\text{Hom}(F_i, F_j)$  is either empty or a singleton depending on whether  $F_i \leq F_j$ .

**Remark 4.1.** Thinness is the categorical formalization of no competing interpretations: for any two failure classes, there exists at most one admissible semantic escalation path. This eliminates ambiguity in regime dominance and ensures that reconciliation is canonical rather than heuristic.

## 5 Policy Compilation Semantics

**Definition 5.1 (Policy Action Set).** Let  $P$  be the set of policy actions (e.g., restrict rollout, freeze control-plane, page on-call, isolate dependency, etc.).

**Definition 5.2 (Policy Monoid).** A policy monoid is a triple  $(P, \oplus, e)$  where  $\oplus$  is associative policy composition and  $e$  is the identity (no-op) policy.

*Proof.* Associativity enables stable bundling;  $e$  ensures explicit “no action” is representable.

**Definition 5.3 (Policy Compiler).** A policy compiler is a deterministic map

$$\Pi : F \rightarrow P$$

that assigns a canonical policy bundle to each failure class.

**Theorem 5.1 (Determinism of Policy Compilation).** The composed function  $\Pi \circ C : O \rightarrow P$  is deterministic.

*Proof.* Composition of deterministic functions is deterministic.

## 6 Formal Semantic Examples

### 6.1 Example 6.1: Single-Regime Classification and Compilation

Let  $o \in O$  be an observation whose invariant-violations correspond to a unique regime  $F_k$  (e.g., Control Plane Saturation). Then:

$$C(o) = F_k, \quad (\Pi \circ C)(o) = \Pi(F_k).$$

Interpretation: the system resolves to one stable semantic regime; compilation emits one canonical policy bundle.

## 6.2 Example 6.2: Multi-Regime Join and Canonical Reconciliation

Let  $o \in O$  trigger evidence consistent with two regimes  $F_a$  and  $F_b$ . If  $(F, \leq)$  admits join:

$$C(o) = F_a \vee F_b.$$

Compiled policy is:

$$\Pi(C(o)) = \Pi(F_a \vee F_b).$$

Interpretation: join selects a unique least-dominating semantic regime (no ambiguity), and compilation produces a single resolved policy.

## 6.3 Example 6.3: Explicit Neutral Outcome (No-Action as a First-Class Result)

If  $o$  violates no encoded invariants, define:

$$C(o) = F_0, \quad \Pi(F_0) = e.$$

Interpretation: non-action is not absence of meaning; it is the meaning of semantic stability.

# 7 Minimal AEGON Policy DSL and JSON Bundle Target

**Remark 7.1.** This section is intentionally minimal: it demonstrates the “language  $\rightarrow$  policy bundle” idea without over-engineering the compiler.

## 7.1 Five-Construct Minimal Policy DSL (Sketch)

We define a minimal DSL with at most five constructs:

- C1: ON <FailureClass> (trigger)
- C2: DO <Action> (emit action)
- C3: WITH <Key=Value> (attach parameters)
- C4: TIER <Name> (capability gate)
- C5: END (close block)

## 7.2 DSL Example Compiling to JSON Policy Bundle

```
TIER pro
ON CONTROL_PLANE_SATURATION
DO freeze_control_plane
WITH window_minutes=30
DO page_oncall
WITH severity=high
END
```

Compiled JSON Bundle (Target):

```
{
  "tier": "pro",
  "on": "CONTROL_PLANE_SATURATION",
  "actions": [
    { "name": "freeze_control_plane",
      "params": { "window_minutes": 30 } },
    { "name": "page_oncall",
      "params": { "severity": "high" } }
  ]
}
```

## 8 Interpretive Remarks on Classification Totality and Semantic Neutrality

**Remark 8.1 (Poset-as-Category View).** Treating  $(F, \leq)$  as a thin category formalizes semantic dominance structurally rather than procedurally. No probabilistic scoring, confidence weighting, or human arbitration is required to determine escalation.

**Remark 8.2 (Policy Neutrality).** The policy monoid admits a neutral element  $e$  representing no action. Neutral outcomes are first-class semantic results, indicating system stability rather than classifier failure.

## 9 Conclusion

The AEGON Algebra expresses a conceptual result: despite unbounded system scale, observable behavior admits a finite deterministic semantic regime. By equipping failure classes with a partial order (and joins when needed) and compiling semantic regimes into monoidal policy bundles, AEGON supports stable interpretation and action generation without heuristics or learning.

## References

- [1] S. Mac Lane, *Categories for the Working Mathematician*, 2nd ed., Springer, 1998.
- [2] AAD Systems (Adrian Diamond), *The ECASM Algebra*.  
<https://papers.aadsystems.com/ecasm-algebra>
- [3] AAD Systems (Adrian Diamond), *AEGON Technical Documentation*, v1.0.  
<https://egon.aadsystems.com/static/documentation.html>