THE ZERO PARADOX

ZP-J AFA Addendum

Decoration Uniqueness from Valuation Structure

Version 1.1 | May 2026

v1.1: Header banner added. v1.0: Initial release. Presents the derivation chain from ValuationStructure to AFA decoration uniqueness for finite Accessible Pointed Graphs. All active theorems sorry-free in Lean 4 (one commented-out stub; see §V). Reads after ZP-J Self-Reference. Axiom footprint: [propext, Classical.choice, Quot.sound] throughout.

This document presents the formal consequences of ZP-J's valuation framework for the central uniqueness theorem of Aczel's Anti-Foundation Axiom (AFA). The main result is decoration_unique (ZPJ_APG.lean §IX): for any finite Accessible Pointed Graph, any two valid decorations must agree at every vertex. The proof does not import set-theoretic AFA axioms. It derives the uniqueness property from a chain of abstract typeclasses whose root is ValuationStructure, established in ZP-J.

Readers of ZP-J Self-Reference will recognise the key move: ⊥ is the unique fixed point of scale because any non-⊥ element has finite depth, and scale increases depth by 1. The same argument, iterated k times, forces every vertex on a directed cycle of length k to carry decoration ⊥. Acyclic vertices are handled by strong induction on the size of the reachable set. Both cases together give decoration_unique.

Section I: ValuationStructure and its Unique Fixed Point

ValuationStructure is the root typeclass of the derivation chain. It abstracts the depth-measure argument that drives every uniqueness result in this document. A type L carries a ValuationStructure when it has a ZPSemilattice structure, a self-application operation scale, and a depth measure val taking values in ℕ∞ (the natural numbers extended with a point at infinity).

Typeclass: ValuationStructure (ZPJ_Scale.lean)
class ValuationStructure (L : Type*) [ZPSemilattice L] where
scale : L → L -- self-application
val : L → ℕ∞ -- depth measure
scale_bot : scale ⊥ = ⊥ -- ⊥ is a fixed point of scale
val_bot : val ⊥ = ∞ -- ⊥ has infinite depth
val_unique : ∀ x, val x = ∞ → x = ⊥ -- infinite depth identifies ⊥

Typeclass: ValuationStructure (ZPJ_Scale.lean)
val_scale : ∀ x ≠ ⊥, val (scale x) = val x + 1 -- scale strictly increases depth

The four axioms are minimal. val_bot and val_scale together give the key consequence: for any x ≠ ⊥, val(x) is finite, and applying scale strictly increases it. No element with finite depth can therefore satisfy scale(x) = x.

Theorem: scale_unique_fp (ZPJ_Scale.lean)
∀ x : L, scale x = x → x = ⊥
⊥ is the only fixed point of scale.
Proof: suppose scale x = x and x ≠ ⊥. By val_scale, val(scale x) = val(x) + 1. But scale x = x gives val(x) = val(x) + 1, which is impossible in ℕ∞ for any finite value. Contradiction.
Lean purity: [propext, Classical.choice, Quot.sound]. ✓

Section II: The Derivation Chain

ValuationStructure generates two further typeclasses by successive derivation. AbstractSelfApp extracts the fixed-point structure from ValuationStructure, replacing scale with an abstract selfApp operation. AFAStructure — the three-field typeclass encoding the Quine atom properties — is then derived from AbstractSelfApp. At each step, the fields of the target typeclass are proved as theorems from the source. No new axioms are introduced. The relationship to Aczel's theorem in ZF+AFA is discussed in Remark R-J.A (§V).

Typeclass: AbstractSelfApp (ZPJ_SelfApp.lean)
class AbstractSelfApp (L : Type*) [ZPSemilattice L] where
selfApp : L → L
fixed_bot : selfApp ⊥ = ⊥
unique_fp : ∀ x : L, selfApp x = x → x = ⊥
Instance toAbstractSelfApp (ZPJ_Scale.lean):
selfApp := scale
fixed_bot := scale_bot (direct from ValuationStructure)
unique_fp := scale_unique_fp (proved in §I above)
No new axioms.

AFAStructure is the lattice-level encoding of the three structural facts that ZF+AFA provides set-theoretically: that ⊥ contains itself, that it is the only self-containing element, and the self-membership predicate itself. In ZF+AFA, the existence of a self-containing set follows from AFA's existence clause applied to the one-node self-loop graph; uniqueness follows from AFA's uniqueness clause. In the ZP encoding, the existence field (bot_self_mem) is supplied directly by fixed_bot from AbstractSelfApp. The uniqueness field (quine_unique) is proved as a theorem from unique_fp — no new axioms are introduced at this step.

Typeclass: AFAStructure (ZPJ.lean)

class AFAStructure (L : Type*) [ZPSemilattice L] where
selfMem : L → Prop
quine_unique : ∀ x y : L, selfMem x → selfMem y → x = y
bot_self_mem : selfMem ⊥
Instance toAFAStructure (ZPJ_SelfApp.lean):
selfMem := λ x, selfApp x = x (fixed-point predicate)
bot_self_mem := fixed_bot (⊥ is a fixed point)
quine_unique := derived from unique_fp (any fixed point is ⊥)
No new axioms.

Proposition: Derivation Chain (ZPJ_Scale.lean, ZPJ_SelfApp.lean)

ValuationStructure L ⇒ AbstractSelfApp L ⇒ AFAStructure L
Each arrow is a Lean instance derivation proved without new axioms.
Any type satisfying ValuationStructure inherits the full AFAStructure as a chain of theorems.
Lean purity: [propext, Classical.choice, Quot.sound]. ✓

Remark: Scope of the Chain

The derivation chain shows that ValuationStructure is sufficient to satisfy AFAStructure's three fields within the ZP typeclass hierarchy. It does not show that AFA is derivable from ZF: Foundation and AFA remain mutually exclusive set-theoretic frameworks. The chain is internal to the ZP lattice abstraction and says nothing about which set-theoretic axioms hold. The precise relationship to Aczel's decoration theorem is discussed in Remark R-J.A (§V).

Section III: Accessible Pointed Graphs and Decorations

An Accessible Pointed Graph (APG) is the combinatorial setting for AFA's central theorem. The decoration uniqueness theorem asserts that any two valid labellings of an APG's vertices must agree. This section

defines both notions in the abstract setting of ZP's DecorationUniverse typeclass.

Definition: Accessible Pointed Graph (ZPJ_APG.lean §I)
An APG over vertex type V (a Quiver) is a structure APG V with:
root : V
accessible : ∀ v : V, Reachable root v
Every vertex is reachable from root by following directed edges.
children(v) = { w : V | v → w } (immediate successors)
Reach(v) = { w : V | Reachable v w } (all vertices reachable from v)
In a finite APG (Fintype V), every Reach(v) is a finite set. |Reach(v)| is the cardinality used in the induction of §IV.

A decoration assigns labels from a DecorationUniverse to each vertex, subject to a local consistency condition: the label at v is assembled from the labels of its immediate successors via the collect operation.

Typeclass: DecorationUniverse (ZPJ_APG.lean §II)
class DecorationUniverse (U : Type*) [ZPSemilattice U]
[ValuationStructure U] where
collect : Set U → U
collect_singleton : ∀ x : U, collect {x} = scale x
collect_val_ge : ∀ (S : Set U) (x : U), x ∈ S → val (collect S) ≥ val x + 1
IsDecoration d means: ∀ v, d v = collect (d “′ children v)
where d “′ children v = { d w | w ∈ children v }
is the image of the successor set under d.

Section IV: Decoration Uniqueness

The main theorem asserts that any finite APG admits at most one valid decoration. The proof splits on whether a vertex lies on a directed cycle.

Theorem: decoration_unique (ZPJ_APG.lean §IX)
∀ {V : Type*} [Quiver V] [Fintype V]

Theorem: decoration_unique (ZPJ_APG.lean §IX)

{U : Type*} [ZPSemilattice U] [ValuationStructure U] [DecorationUniverse U]

(G : APG V) (d_1 d_2 : V → U),

IsDecoration d_1 → IsDecoration d_2 → $d_1 = d_2$

For any finite APG, any two valid decorations into a DecorationUniverse agree at every vertex.

Lean purity: [propext, Classical.choice, Quot.sound]. ✓

IV.1 — Cyclic Vertices

A vertex v is cyclic if there exists a directed path from v back to itself. The valuation argument forces any valid decoration to assign ⊥ to every cyclic vertex, independent of which decoration is used. Both d_1 and d_2 must therefore assign ⊥ to every cyclic vertex, and they agree trivially on this case.

The key tool is the iterated valuation lemma. If v lies on a cycle of length k ≥ 1, the decoration equation applied k times around the cycle gives d(v) = scale□(d(v)). For any x ≠ ⊥, val(scale□(x)) = val(x) + k (val_iterate), so a fixed point would require val(x) = val(x) + k — impossible for finite val. Therefore d(v) = ⊥.

Lemma: val_iterate (ZPJ_APG.lean §III)

∀ (x : U) (hx : x ≠ ⊥) (k : ℕ),

val (scale^[k] x) = val x + k

For any x ≠ ⊥, applying scale k times increases depth by exactly k.

Proof: induction on k; val_scale applies at each step because scale^[n](x) ≠ ⊥ follows from the induction hypothesis and finiteness of val(x).

Lean purity: [propext, Classical.choice, Quot.sound]. ✓

Lemma: scale_iterate_unique_fp (ZPJ_APG.lean §IV)

∀ (k : ℕ) (hk : 0 < k) (x : U), scale^[k] x = x → x = ⊥

⊥ is the only element fixed by any k-fold iteration of scale (k ≥ 1).

Proof: if scale^[k] x = x and x ≠ ⊥, then val_iterate gives val(x) = val(x) + k with k ≥ 1 — contradiction.

Lean purity: [propext, Classical.choice, Quot.sound]. ✓

Theorem: cyclic_decoration_eq_bot (ZPJ_APG.lean §VII')

∀ (d : V → U), IsDecoration d →

(v lies on a directed cycle) → d v = ⊥

Theorem: cyclic_decoration_eq_bot (ZPJ_APG.lean §VII')
Consequence: $d_1\ v = d_2\ v = \bot$ for every cyclic vertex v.
Lean purity: [propext, Classical.choice, Quot.sound]. ✓

IV.2 — Acyclic Vertices

A vertex v is acyclic if no directed cycle passes through it. The argument uses strong induction on |Reach(v)|. Every vertex reaches itself via the empty path, so |Reach(v)| ≥ 1 for every v; the n = 0 branch is vacuous. All actual vertices fall into the inductive step.

Inductive step. Suppose d_1 and d_2 agree on every vertex w with |Reach(w)| < n, and suppose |Reach(v)| = n with v acyclic. For each successor w ∈ children(v): since v is acyclic, v is not reachable from w, so Reach(w) ⊊ Reach(v), giving |Reach(w)| < n. By the induction hypothesis, $d_1(w) = d_2(w)$ for every w ∈ children(v). When children(v) = ∅ this hypothesis is vacuously satisfied: collect is a function, the image of ∅ under any decoration is ∅, so both $d_1(v)$ and $d_2(v)$ equal collect(∅) and agree. When children(v) ≠ ∅, the induction hypothesis gives d_1 “′ children(v) = d_2 “′ children(v), and the decoration equation then gives $d_1(v)$ = collect(d_1 “′ children(v)) = collect(d_2 “′ children(v)) = $d_2(v)$. In both sub-cases, acyclic_induction_step formalises the argument.

Lemma: acyclic_induction_step (ZPJ_APG.lean §VIII)
If d_1 and d_2 agree on every successor of an acyclic vertex v,
then $d_1\ v = d_2\ v$.
Proof: collect is the same operation for both; d_1 and d_2 agree pointwise on children(v) by hypothesis; therefore the assembled labels agree.
Lean purity: [propext, Classical.choice, Quot.sound]. ✓

IV.3 — Combining the Cases

Every vertex in a finite APG is either cyclic or acyclic. The two cases are exhaustive and jointly establish $d_1(v) = d_2(v)$ for every vertex v. Since V is a Fintype, funext closes the global equality $d_1 = d_2$.

Section V: Scope, Purity, and Open Questions

decoration_unique establishes the uniqueness half of AFA's central theorem for abstract DecorationUniverses over finite graphs. Two scope boundaries are worth stating explicitly.

Existence. This document does not prove that every finite APG admits a valid decoration. Uniqueness and existence are independent: one can show that no two decorations can differ without showing that any decoration exists. The existence half is not formalised here for abstract DecorationUniverses and remains an open question.

Finite graphs. decoration_unique requires Fintype V. The strong induction on |Reach(v)| terminates because V is finite. Extending the result to infinite APGs would require an ordinal induction or a well-foundedness argument on the reachability relation, and is not addressed here.

Commented-out stub. ZPJ_APG.lean contains a commented-out theorem acyclic_decoration_unique with a sorry placeholder. That stub is not used by the proof of decoration_unique: the final proof proceeds by direct strong induction using acyclic_induction_step, without using the stub. All active (non-commented-out) theorems in the seven source files are sorry-free.

Lean Source Files
ZPJ_Scale.lean — ValuationStructure, scale_unique_fp, toAbstractSelfApp
ZPJ_SelfApp.lean — AbstractSelfApp, derived_bot_self_mem, derived_quine_unique, toAFAStructure
ZPJ_AczelConn.lean — J_self, selfMem_determines_singleton, DC-free identification theorems
ZPJ_OntBridge.lean — OntologicalStates as AbstractSelfApp instance
ZPJ_Model.lean — $\mathbb{N}\infty$ as ValuationStructure instance
ZPJ_ScaleBridge.lean — $\mathbb{Z}_2$ as ValuationStructure instance (via ValBridge typeclass)
ZPJ_APG.lean — APG, DecorationUniverse, val_iterate, scale_iterate_unique_fp, cyclic_decoration_eq_bot, acyclic_induction_step, decoration_unique
All files in ZeroParadox/ in the public repository.

Axiom Footprint (all results in this document)
[propext, Classical.choice, Quot.sound]
propext — propositional extensionality (standard in Lean 4)
Classical.choice — choice principle (Mathlib Finset and Fintype machinery)
Quot.sound — quotient soundness (standard in Lean 4)
No ZP-specific axioms beyond [propext, Classical.choice, Quot.sound].
No Dependent Choice. No additional set-theoretic assumptions.

Remark R-J.A — Relationship to Aczel's Theorem
Aczel's decoration theorem (Non-Well-Founded Sets, CSLI 1988) states that every APG has a unique decoration into the universe of non-well-founded sets. The result here is not a re-proof of Aczel's theorem by different methods. The objects are different: Aczel's target is a specific set-theoretic universe; the target here is any type carrying ValuationStructure and a collect operation, with no set-membership semantics required. The contribution is the generalisation: decoration uniqueness holds for any abstract DecorationUniverse satisfying the depth-measure axioms, independently of set-theoretic content. Whether the existence half of Aczel's theorem generalises to abstract DecorationUniverses is an open question.

Endnote: This document is an addendum to ZP-J Self-Reference and reads after it. The derivation chain (ValuationStructure → AbstractSelfApp → AFAStructure) is established in ZP-J; this document applies it to the APG decoration problem. Version 1.0 covers the uniqueness result for finite graphs. All active theorems sorry-free in Lean 4 as of May 2026 (one commented-out stub; see §V).