THE ZERO PARADOX

ZP-J AFA Addendum

Decoration Uniqueness and Existence from Valuation Structure

Initial: May 2026 | Current: 1.25, November 2026

This document presents the formal consequences of ZP-J's valuation framework for the central uniqueness CLAUSE of Aczel's Anti-Foundation Axiom (AFA). The main result is decoration_unique_of_val (ZeroParadox/Settheory/APG.lean § IX): any two valid decorations of any quiver, finite or not, into a DecorationUniverse are equal, with no APG argument and no finiteness hypothesis; it measures [propext, Quot.sound]. decoration_unique (same section) is the finite, graph-side proof: for any finite Accessible Pointed Graph, any two valid decorations must agree at every vertex. Its statement is a special case of the main result (the example after decoration_unique_finite in the same section). Neither proof imports set-theoretic AFA axioms. Each derives the uniqueness property from a chain of abstract typeclasses whose root is ValuationStructure, established in ZP-J. The analogue of AFA's existence clause is proved too (§IV.5): every quiver has exactly one valid decoration into every DecorationUniverse (existsUnique_decoration), measuring [propext, Quot.sound].

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. Depth is what settles the cyclic case too, though by bounding rather than by iterating: going once around a cycle forces a vertex's depth to exceed itself, which only infinite depth — that is, ⊥ — can do. Acyclic vertices are handled by strong induction on the size of the reachable set. Both cases together give decoration_unique, the graph-side proof. Read from the values instead of the vertices, the same depth does all the work and finiteness none: a vertex whose value has finite depth has children whose values have smaller depth, and the only value of infinite depth is ⊥ of U. That is the main result (§IV.4).

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 (ZeroParadox/Valuation/Scale.lean)
class ValuationStructure (L : Type*) [ZPSemilattice L] where
scale : L → L
val : L → ℕ∞
scale_bot : scale bot = bot
val_bot : val bot = ⊤
val_unique : ∀ x : L, val x = ⊤ → x = bot
val_scale : ∀ x : L, x ≠ bot → val (scale x) = val x + 1
Transcribed from ZeroParadox/Valuation/Scale.lean. ⊤ is the top of ℕ∞, written ∞ in the prose above; bot is ZPSemilattice's ⊥. Read: scale fixes the bottom, the bottom has infinite depth, only the bottom does, and scale adds one everywhere else.

val_unique and val_scale carry this section's argument: val_unique gives, for any x ≠ ⊥, that val(x) is finite — it is the contrapositive of "infinite depth identifies ⊥" — and val_scale makes scale strictly increase it. No element of finite depth can therefore satisfy scale(x) = x. val_bot is consumed nowhere on that chain (ZeroParadox/Valuation/Scale.lean, § I); it fixes the depth of ⊥ itself, which the fixed-point argument never reads.

Theorem: scale_unique_fp (ZeroParadox/Valuation/Scale.lean)
∀ x : L, ValuationStructure.scale x = x → x = bot
⊥ 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 has no finite solution in ℕ∞, and val(x) is finite by val_unique. So scale x ≠ x for every x ≠ ⊥ (scale_ne_fixed), a 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 LAWS of the target typeclass are proved as theorems from the source, and the DATA fields are supplied by definition: the selfApp field of AbstractSelfApp is assigned (selfApp := scale), and the selfMem field of AFAStructure is supplied by def selfMemDerived. TWO of the three AFAStructure fields become theorems, not three (ZeroParadox/Computability/SelfApp.lean, def selfMemDerived). No new axioms are introduced. The relationship to Aczel's Anti-Foundation Axiom in

ZF+AFA is discussed in Remark R-J.A (§V).

Typeclass: AbstractSelfApp (ZeroParadox/Computability/SelfApp.lean)
class AbstractSelfApp (L : Type*) [ZPSemilattice L] where
selfApp : L → L
fixed_bot : selfApp bot = bot
unique_fp : ∀ x : L, selfApp x = x → x = bot
Instance toAbstractSelfApp (ZeroParadox/Valuation/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 ⊥ satisfies selfMem, read in ZF+AFA as x = {x} (bot_self_mem), that it is the only element that does (quine_unique), and the predicate selfMem itself. In ZF+AFA, the existence of a set x = {x} 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 a LAW, discharged by a theorem built from fixed_bot in AbstractSelfApp. The uniqueness field (quine_unique) is likewise a law, proved from unique_fp. Only selfMem is data, and only it is supplied — no new axioms are introduced at this step.

Typeclass: AFAStructure (ZeroParadox/Settheory/SetTheoryAFA.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 bot
Instance toAFAStructure (ZeroParadox/Computability/SelfApp.lean):
selfMem := selfMemDerived (DATA - supplied, a def)
bot_self_mem := derived_bot_self_mem (LAW - discharged by a theorem)
quine_unique := derived_quine_unique (LAW - discharged by a theorem)
No new axioms. Two of the three fields are laws and become theorems;
selfMem is data and is supplied.

Proposition: Derivation Chain (ZeroParadox/Valuation/Scale.lean, ZeroParadox/Computability/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 two AFAStructure LAWS as theorems; its data field selfMem is supplied by definition, not proved.
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 Anti-Foundation Axiom 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 the unique-decoration property AFA asserts — an AXIOM, not a theorem; see Remark R-J.A. 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 (ZeroParadox/Settheory/APG.lean §I)
An APG over vertex type V (a Quiver) is a structure APG V with:
root : V
accessible : ∀ v : V, Nonempty (Quiver.Path root v)
Every vertex is reachable from root by following directed edges.
children(v) := apg_children v = { w | Nonempty (v ⟶ w) } (immediate successors)
Reach(v) = { w : V | Nonempty (Quiver.Path v w) } (reachable from v)
In a finite APG (Fintype V), every Reach(v) is a finite set. |Reach(v)| is the cardinality used in the graph-side induction of ZeroParadox/Settheory/APG.lean § IX (this document's §IV.2); the value-side proof (§IV.4) uses no cardinality.

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 (ZeroParadox/Settheory/APG.lean §II)
class DecorationUniverse (U : Type*) [ZPSemilattice U] [ValuationStructure U] where
collect : Set U → U
collect_singleton : ∀ x : U, collect {x} = ValuationStructure.scale x
collect_val_ge : ∀ (S : Set U) (x : U), x ∈ S →
ValuationStructure.val (collect S) ≥ ValuationStructure.val x + 1
Transcribed from ZeroParadox/Settheory/APG.lean. IsDecoration d means ∀ v, d v = collect (d '' apg_children v), where d '' S is Lean's set image — { d w | w ∈ S }. • The two axioms pin only the SINGLETON case and a lower bound; they do not require collect to assemble a parent's value from its children's.

Section IV: Decoration Uniqueness and Existence

The main theorem, decoration_unique_of_val (§IV.4), asserts that any two valid decorations of any quiver, finite or not, are equal; its proof splits on whether the decorating value has finite depth. §§IV.1–IV.3 give the finite, graph-side proof, decoration_unique: any finite APG admits at most one valid decoration, by splitting on whether a vertex lies on a directed cycle. Its statement is a special case of the main theorem. §IV.5 proves existence: every quiver has a valid decoration.

Proposition: decoration_unique (ZeroParadox/Settheory/APG.lean §IX)
∀ {V : Type*} [Quiver V] [Fintype V]
{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 non-empty directed path from v back to itself (HasSelfCycle: an edge out of v followed by a path back). 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 argument is a chain of two inequalities, not an iterated equation. Let w be the next vertex on the cycle. Because w is a child of v and d(v) collects over d of v's children, collect_val_ge gives val(d(v)) ≥ val(d(w)) + 1. Following the cycle back from w to v, path_val_chain gives val(d(w)) ≥ val(d(v)) + length(p). Together these force val(d(v)) to exceed itself, which no finite value in ℕ∞ can do, so val(d(v)) = ∞ and therefore d(v) = ⊥.

Lemma: val_iterate (ZeroParadox/Settheory/APG.lean § III)
∀ {U : Type*} [ZPSemilattice U] [ValuationStructure U] (x : U),
x ≠ ZPSemilattice.bot → ∀ (k : ℕ),
ValuationStructure.val (ValuationStructure.scale^[k] x) =
ValuationStructure.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).
• This is a genuine lemma of § III and it drives scale_iterate_unique_fp below. It is NOT what the cyclic case above uses: cyclic_decoration_eq_bot never forms scale^[k].
Lean purity: [propext, Classical.choice, Quot.sound]. ✓

Lemma: scale_iterate_unique_fp (ZeroParadox/Settheory/APG.lean §IV)
∀ {U : Type*} [ZPSemilattice U] [ValuationStructure U] (k : ℕ),
0 < k → ∀ (x : U), ValuationStructure.scale^[k] x = x →
x = ZPSemilattice.bot
⊥ 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 (ZeroParadox/Settheory/APG.lean § VII′)
∀ {V : Type*} [Quiver V] {U : Type*} [ZPSemilattice U]
[ValuationStructure U] [DecorationUniverse U] (d : V → U),
IsDecoration d → ∀ (v : V), HasSelfCycle v → d v = ZPSemilattice.bot
Consequence: $d_1\ v = d_2\ v = \bot$ for every cyclic vertex v.
Proof route, as written: collect_val_ge across one edge gives val(d v) ≥ val(d w) + 1, path_val_chain around the cycle gives val(d w) ≥ val(d v) + length(p), and the two together are unsatisfiable for finite val. It consumes val_unique through val_finite_of_ne_bot, and assumes nothing about how many children a vertex has.
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 (ZeroParadox/Settheory/APG.lean §VIII)
If d_1 and d_2 agree on every successor of a vertex v (no acyclicity hypothesis),
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, 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. Function extensionality then closes the global equality $d_1 = d_2$ from that pointwise agreement; finiteness is consumed earlier, by the acyclic case's descent on |Reach(v)|, and plays no part in this last step.

IV.4 — The Value Side: Any Quiver

The same uniqueness holds with no finiteness at all, by inducting on the decorating value rather than on the graph. Fix a vertex u. If $\mathrm{val}(d_1\ u)$ is a natural number m, then for each child w, collect_val_ge gives $m \geq \mathrm{val}(d_1\ w) + 1$, so $\mathrm{val}(d_1\ w)$ is a natural number below m; strong induction on m and acyclic_induction_step give $d_1\ u = d_2\ u$. If $\mathrm{val}(d_1\ u)$ and $\mathrm{val}(d_2\ u)$ are both $\top$, val_unique makes both equal $\bot$ of U; if only $\mathrm{val}(d_2\ u)$ is finite, the finite case applies with the roles swapped. No cardinality is counted, so the graph may be infinite.

Theorem: decoration_unique_of_val (ZeroParadox/Settheory/APG.lean §IX)
$\forall$ {V : Type*} [Quiver V]
{U : Type*} [ZPSemilattice U] [ValuationStructure U] [DecorationUniverse U]
$(d_1\ d_2 : V \to U)$,
IsDecoration $d_1 \to$ IsDecoration $d_2 \to d_1 = d_2$
On every quiver, finite or not, any two valid decorations into a DecorationUniverse are equal. No APG argument, no finiteness binder.
Prior art: the AFA2 half of a result Aczel sets as an exercise (Non-Well-Founded Sets, CSLI 1988, Ex. 2.1 p. 20; AFA1/AFA2 p. 19), proved here by strong induction on the finite valuation (depth) of the decorating value, adapting the argument Aczel indicates on p. 19 that equality between well-founded sets is uniquely determined, by transfinite induction on the membership relation. One difference from the exercise is a point of valuation $\top$, $\bot$ of U, admitted through val_unique.
Lean purity: [propext, Quot.sound]. ✓

Corollary: decoration_unique_finite (ZeroParadox/Settheory/APG.lean §IX)

decoration_unique with [Finite V] in place of [Fintype V], obtained from decoration_unique_of_val; the [Finite V] binder appears in the statement only.

Lean purity: [propext, Quot.sound]. ✓

The two proofs are two charts of one uniqueness, and neither is the structure. The graph side splits vertices cyclic/acyclic and spends finiteness on |Reach(v)|; the value side splits depth finite/⊤ and spends none. They are related one way: a cyclic vertex's value has depth ⊤ (cyclic_decoration_eq_bot with val_bot); no converse is claimed.

IV.5 — Existence: Any Quiver

Uniqueness has a partner: every quiver has a valid decoration into every DecorationUniverse, so it has exactly one. On a vertex that is accessible for the child relation ChildRel (Lean's Acc, the inductive form of well-foundedness below the vertex), the value is defined by recursion on that accessibility: collect of the children's values (accDec). On every other vertex it is ⊥ of U. The construction never asks whether a vertex is accessible. It collects every value a child is related to (ValRel: an accessible child to its recursive value, any other child to ⊥ of U), and the proof then splits on the valuation of the collected value, which is a natural number or ⊤, so no case is decided by Classical.choice.

Theorem: existsUnique_decoration (ZeroParadox/Settheory/APG.lean §X)

∀ {V : Type*} [Quiver V]

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

∃! d : V → U, IsDecoration d

Every quiver, finite or not, has exactly one valid decoration into a DecorationUniverse: exists_decoration gives existence, decoration_unique_of_val uniqueness.

Prior art: AFA1, "every graph has at least one decoration" (Aczel, Non-Well-Founded Sets, CSLI 1988, p. 19); CRA-existence and CRA-uniqueness for an algebra (Capretta, Uustalu and Vene, SBMF 2009, LNCS 5902, Def. 1 p. 86); both halves for one ranking algebra (Urabe, Hara and Hasuo, arXiv:1704.04872, Lemma A.6 p. 15). The abstraction over carriers was not located as of 2026-11-10, searched in those papers, theorem-search engines, Lean libraries and the pinned Mathlib.

Lean purity: [propext, Quot.sound]. ✓

Proposition: decoration_eq_bot_of_not_acc (ZeroParadox/Settheory/APG.lean §X)

∀ (d : V → U), IsDecoration d → ∀ v, ¬ Acc (ChildRel V) v → d v = ZPSemilattice.bot

One way only: a vertex that is not accessible is decorated ⊥ of U. The converse fails: in the fan, a root with a child of every finite height, the root is accessible and every decoration still sends it to ⊥ of U (fan_root_acc_and_eq_bot).

Lean purity: [propext, Quot.sound]. ✓

Section V: Scope and Purity

decoration_unique_of_val establishes the uniqueness half of the property AFA asserts (an axiom — Remark R-J.A) for abstract DecorationUniverses over every quiver, and decoration_unique by a graph-side proof over finite graphs; exists_decoration establishes the existence half over every quiver (§IV.5). The scope boundary, and where finiteness enters, are stated below.

Existence. Every quiver has a valid decoration into every DecorationUniverse: exists_decoration (ZeroParadox/Settheory/APG.lean §X; this document's §IV.5).

Finiteness. decoration_unique requires Fintype V: its strong induction on |Reach(v)| terminates because V is finite. Uniqueness itself does not require it: decoration_unique_of_val (§IV.4) holds on every quiver, finite or infinite, by induction on the decorating value's depth.

Commented-out stub. ZeroParadox/Settheory/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 eight source files are sorry-free.

Lean Source Files
ZeroParadox/Valuation/Scale.lean — ValuationStructure, scale_unique_fp, toAbstractSelfApp
ZeroParadox/Computability/SelfApp.lean — AbstractSelfApp, selfMemDerived, derived_bot_self_mem, derived_quine_unique, toAFAStructure
ZeroParadox/Settheory/SetTheoryAFA.lean — AFAStructure typeclass (selfMem, quine_unique, bot_self_mem), IsQuineAtom
ZeroParadox/Settheory/AczelConn.lean — J_self, selfMem_determines_singleton, choice-free identification theorems
ZeroParadox/Settheory/OntBridge.lean — OntologicalStates as AbstractSelfApp instance
ZeroParadox/Settheory/Model.lean — instNatInfZPS and instNatInfVal, ℕ∞ as a ValuationStructure instance
ZeroParadox/Valuation/ScaleBridge.lean — instZ2ValBridge : ValBridge ℤ_[2]. Note no ValuationStructure ℤ_[2] is REGISTERED, nor any ZPSemilattice ℤ_[2] — which is a fact about what is declared, not about what is possible: ZeroParadox/Valuation/Scale.lean § V builds one and discharges all four axioms over it. ValBridge is the variant that drops the semilattice requirement while keeping the same four axioms.
ZeroParadox/Settheory/APG.lean — APG, DecorationUniverse, val_iterate, scale_iterate_unique_fp, cyclic_decoration_eq_bot, acyclic_induction_step, decoration_unique, decoration_unique_of_val, decoration_unique_finite, exists_decoration, existsUnique_decoration, decoration_eq_bot_of_not_acc, fan_root_acc_and_eq_bot
All paths are repository-relative, in the public repository.

Axiom Footprint — measured per declaration, not asserted document-wide
[propext, Classical.choice, Quot.sound]
propext — propositional extensionality (standard in Lean 4)
Classical.choice — AT LEAST FOUR ROUTES, and they are not the same claim. (a) THE STATEMENT: decoration_unique binds [Fintype V], and Fintype carries the axiom under the instances the pinned Mathlib supplies, so the statement reports it before any proof and is STATEMENT-CARRIED in the sense of ZeroParadox/Category/ChoiceCannotBe.md (the statement controls are the ZeroParadox/Settheory/APG.lean record below). ValuationStructure and DecorationUniverse themselves report no axioms. (b) A DIRECT CALL: decoration_unique's proof takes an edge out of a Nonempty via hw.some, and Nonempty.some IS Classical.choice — not inherited through packaging, invoked. (c) CARDINALITY: the termination measure runs on Set.ncard, and Set.ncard_lt_ncard and Set.ncard_pos are each independently choice-tainted. (d) THE CASE SPLIT: the proof's by_cases on HasSelfCycle u is decided by Classical.propDecidable, the one instance in the gateway triple (with Classical.indefiniteDescription and Classical.ofNonempty) that L. J. Close identifies in Mathlib's choice dependence (The Axiom Profile of Mathlib, Zenodo 2026, doi:10.5281/zenodo.20710921, pp. 5–6, measured at Mathlib 8f9d9cf on Lean v4.28.0, not the Mathlib 56a0058 on Lean v4.35.0 this repository pins; cited, not claimed).
Quot.sound — quotient soundness (standard in Lean 4)
• NO PROOF CAN CLEAR IT FOR THE THEOREM AS STATED, AND THE STATEMENT IS READ UNDER THE PINNED INSTANCES — both halves are the claim. The record is the DecorationUniverse [ZP-CUSTOM] line of ZeroParadox/Settheory/APG.lean: the full statement of decoration_unique, assumed and used for nothing, reports the three axioms; the same statement with [Finite V] in place of [Fintype V] reports none; and ∀ {U} [ZPSemilattice U] [ValuationStructure U], True reports none. So the [Fintype V] binder carries the axiom and the [ValuationStructure U] binder does not. Which bin a statement falls in is relative to the instances it is elaborated under (ZeroParadox/Category/ChoiceCannotBe.lean § III, STATEMENT-CARRIED IS INSTANCE-RELATIVE). Routes (b), (c) and (d) lie in the proof. The [Finite V] restatement has a proof free of all four: decoration_unique_finite, from decoration_unique_of_val, each [propext, Quot.sound] (§IV.4).
The existence results of §IV.5 (exists_decoration, existsUnique_decoration, decoration_eq_bot_of_not_acc) measure [propext, Quot.sound]: none of the four routes occurs in them.
No ZP-specific AXIOM is declared anywhere in this chain; the footprint above is what the Lean reports, not a commitment the framework makes.
No Dependent Choice. No additional set-theoretic assumptions.

Remark R-J.A — Relationship to Aczel's Anti-Foundation Axiom

Aczel’s ANTI-FOUNDATION AXIOM (Non-Well-Founded Sets, CSLI 1988, ch. 1 p. 6) states that every GRAPH has a unique decoration into the universe of sets — an axiom, not a theorem, and quantified over graphs rather than over accessible pointed ones. (The book’s decoration THEOREM is Mostowski’s Collapsing Lemma, for WELL-FOUNDED graphs.) The result here is not a re-proof of Aczel's axiom 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. So decoration uniqueness holds for any abstract DecorationUniverse satisfying the depth-measure axioms, independently of set-theoretic content. Read from the target (a reading, as ZeroParadox/Settheory/APG.lean labels it), this is CRA-uniqueness for the algebra collect (Capretta, Uustalu and Vene, Corecursive Algebras: A Study of General Structured Corecursion, SBMF 2009, LNCS 5902, Def. 1 p. 86); Urabe, Hara and Hasuo (Categorical Liveness Checking by Corecursive Algebras, arXiv:1704.04872, Def. II.16 p. 5) use a corecursive algebra as a classifier for (non-)well-foundedness in their ranking domains, classically ranking functions into $\mathbb{N} \cup \{\infty\}$, and prove both halves for one such ranking algebra, into the ordinals up to a cap z, uniqueness by transfinite induction on values below z (Lemma A.6, p. 15); here a cyclic vertex’s value has valuation $\top$ (cyclic_decoration_eq_bot with val_bot). The existence half has its analogue too: every quiver has a decoration into every DecorationUniverse (exists_decoration, §IV.5), AFA1’s shape and CRA-existence for collect.

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. It covers the uniqueness result for every quiver (decoration_unique_of_val), its finite, graph-side proof for APGs (decoration_unique), and existence for every quiver (exists_decoration). All active theorems are sorry-free in Lean 4 (one commented-out stub; see §V).