THE ZERO PARADOX

ZP-J: Executability of Self-Reference

Initial: April 2026 | Current: 2.22, November 2026

This document establishes Theorem T-EXEC (Executability of Self-Reference): in any ZPSemilattice (ZP-A's join-semilattice with least element, axioms A1–A4) carrying the AFAStructure typeclass, whatever fills the Quine-atom role — the lattice-level encoding of the set Q = {Q} of ZF+AFA (Anti-Foundation Axiom) — is provably the bottom element ⊥. This bridges the set-theoretic and order-theoretic layers of the framework at the level of that role; the set identity itself stays in ZF+AFA. CC-1 from ZP-A, which stated "$S_0 = \bot$" as a modelling commitment, is restated: given the typeclass, starting at a Quine atom and starting at ⊥ are the same condition. It is not forced: on a carrier with a second point a valid state sequence can start elsewhere (ZeroParadox/Settheory/OntBridge.lean).

This layer extends the core T-EXEC result in four directions: it identifies the set of self-containing elements as {⊥} with no Classical.choice, and states why that does not bear on Aczel's question about Dependent Choice (Section VII); it generalises the AFAStructure typeclass into an abstraction chain reaching down to a pure valuation structure (Section VIII); it instantiates that chain on two concrete types (Section IX); and it proves global decoration uniqueness and existence, into abstract DecorationUniverses, for every quiver, finite or not, with uniqueness also by a finite, graph-side proof for finite accessible pointed graphs (Section X). All results are sorry-free in Lean 4.

Section I: The Open Question — CC-1 as a Modelling Commitment

I. CC-1 in ZP-A

ZP-A Conditional Claim CC-1 states: if the state sequence is initialised at ⊥, then $\bot \leq S(n)$ for all n. This follows trivially from ZP-A T2 (⊥ is the global minimum), so its formal content is essentially: we are choosing ⊥ as the initial state S_0.

The label "Conditional Claim" (CC) marks this as a modelling commitment — an explicit choice not derivable from the axioms A1–A4 alone. ZP-A's axioms give us a semilattice with a bottom element. They do not say which instantiation of the semilattice should start at that bottom. CC-1 asserts: ours does. This is well-motivated, but it is a choice.

The question ZP-J investigates: is this choice forced? Is there a structural reason — derivable from the framework's foundational commitments — that any well-grounded instantiation of ZP-A must begin at ⊥? The answer is that the choice is restated: if the lattice carries AFAStructure, a sequence starts at a Quine atom exactly when it starts at ⊥ (Section V). It is not forced: on a carrier with a second point a valid state sequence starts above ⊥, at a point that is not a Quine atom (ZeroParadox/Settheory/OntBridge.lean). The commitment to start there remains; it is now expressed through the Quine-atom role rather than as a bare choice of element.

II. The Implicit Identification

ZP-E's DA-1 Path 1 already invokes the Quine atom (CC-2): the role that, under AFA, the set $Q = \{Q\}$ plays as the unique solution of $x = \{x\}$. That the bottom element fills this role was present informally but never formally bridged to CC-1. It appeared as motivation for why $\bot$ is the right starting point, not as a derivation of it.

The gap: ZP-A's lattice order is abstract (defined by axioms A1–A4). AFA is a set-theoretic axiom. There is no automatic connection between "x is self-containing" and "x is the lattice bottom." Connecting them requires a bridge — and that bridge was the missing piece. ZP-J provides it.

> Open question entering ZP-J: Is CC-1 ($S_0 = \bot$) forced by the framework's foundational structure, or is it an independent modelling choice? If forced, what is the structural reason? Answer: restated (ZP-J t_exec_iff), and not forced (a countermodel). Given an AFAStructure, "starts at a Quine atom" and "starts at $\bot$" are equivalent (t_exec_iff), so the starting point is named by its role; the choice to start there remains, since on a carrier with a second point a valid sequence starts elsewhere. The Quine atom (CC-2) is a role, and in ZF+AFA the set $Q = \{Q\}$ plays it; the Lean states the role and its occupant chart by chart, and writes no equation between $\bot$ and Q.

Section II: AFA Machinery — Self-Membership and the Quine Atom

I. The Anti-Foundation Axiom

Standard set theory (ZF) includes the Foundation Axiom: every non-empty set S contains an element disjoint from S. A consequence is that no set can contain itself: Foundation rules out $x \in x$ in ZF, excluding self-referential sets by fiat. In Lean, Mathlib's ZFSet, a model of ZFC, contains no such set (no_quine_atom), because its membership relation is well-founded.

The Anti-Foundation Axiom (AFA, Aczel 1988, ch. 1 p. 6) replaces Foundation with an axiom asserting existence and uniqueness: every graph has a unique decoration. As a consequence, every accessible pointed graph (APG) is a picture of a unique set. Under AFA, self-containing sets are not only possible; the equation $x = \{x\}$ has a unique solution (Section II). The resulting theory ZF+AFA is consistent and expressive — it is the natural set-theoretic home for fixed-point and self-referential structures.

II. The Quine Atom

Under AFA, the equation $x = \{x\}$ has a unique solution. This solution is called the Quine atom, denoted Q. Q is the unique set that contains itself as its sole member: $Q = \{Q\}$. It is not the empty set ($\varnothing = \{\}$ contains nothing); it is not any well-founded set (those cannot contain themselves). Q is the minimal non-trivial AFA set — it contains exactly one thing, and that thing is itself.

The AFA uniqueness guarantee is the key property: there is at most one Quine atom. Any two sets that are each their own sole member are equal. This means the property $x = \{x\}$ uniquely identifies an element — a fingerprint that belongs to exactly one object in the universe. Bare self-membership does not: under AFA the set $0^* = \{\varnothing, 0^*\}$ also contains itself (Aczel 1988, Example 1.5). The framework's requirement in AFA's profile is QuineAtomHost (ZeroParadox/Settheory/QuineHost.lean § VII); QuineHost, whose uniqueness is among self-members, is a different class, which AFA's membership fails, since $Q = \{Q\}$ (Aczel's Ω) and 0^* are

distinct self-members (Aczel 1988, Examples 1.3 and 1.5, pp. 6–7).

AFA Uniqueness (AFAStructure.quine_unique)
For any type L with AFA structure: if x, y ∈ L both satisfy selfMem(x) and selfMem(y), then x = y.
Informally: the Quine atom is unique. The selfMem role (read in ZF+AFA as x = {x}, not bare x ∈ x) is held by at most one element of any AFA-structured type.
Lean: AFAStructure.quine_unique — encoded as a typeclass field, not a theorem, because it does not follow from the other two fields: on the two-state lattice MachinePhase, the selfMem true of every element meets bot_self_mem while two distinct elements satisfy it (quine_unique_independent). It is a hypothesis every theorem here carries, and (Section III.II) any lattice can meet it by a trivial choice of selfMem.

III. Self-Membership as a Lattice Predicate

In ZP-J, the AFA machinery is abstracted minimally. We do not need the full apparatus of accessible pointed graphs, bisimulation, or set decoration. We need only two things: a predicate selfMem on the lattice elements, and the guarantee that it is held by at most one element (quine_unique). The third structural field — bot_self_mem — is where the bridge is built.

The AFAStructure typeclass in ZeroParadox/Settheory/SetTheoryAFA.lean captures exactly this. Its three fields are sufficient to derive T-EXEC with no additional axioms. The full AFA machinery (APGs, bisimulation, decoration) motivates the fields; the formal derivation requires only the fields themselves, and every ZPSemilattice supplies them (Section III.II).

Section III: AFAStructure — The Structural Bridge

I. The Three Fields

The AFAStructure typeclass for a ZP-A semilattice L has three fields. The first two encode standard AFA properties. The third is the bridge.

AFAStructure Typeclass (ZeroParadox/Settheory/SetTheoryAFA.lean § I)
class AFAStructure (L : Type*) [ZPSemilattice L] with:
(1) selfMem : L → Prop — the self-containment predicate. Read in ZF+AFA, selfMem(x) is x = {x} (x is its own sole member), not bare membership x ∈ x: under AFA the set 0* = {∅, 0*} contains itself and is not Q (Aczel 1988, Example 1.5), so the uniqueness below would fail for membership.
(2) quine_unique : ∀ x y : L, selfMem(x) → selfMem(y) → x = y — AFA uniqueness. At most one element of L plays the self-containment role.
(3) bot_self_mem : selfMem(⊥) — the bridge field. The bottom element of the lattice is self-containing. This encodes the Quine-atom ROLE (CC-2), which in ZF+AFA the set Q = {Q} plays; an equation between ⊥ of L and that set would compare an element of L with a set, so it is not stated here.

II. What bot_self_mem Says

bot_self_mem is the single structural claim that connects the order-theoretic world (ZP-A's lattice) to the set-theoretic world (AFA). It says: the bottom element ⊥ of the lattice is self-containing — it satisfies selfMem(⊥).

In ZF+AFA the Quine atom (CC-2) is the set Q = {Q}, its own sole member. The lattice and the set universe are separate charts, so no equation ⊥ = Q is written. What is shared is the ROLE, a point that steps to itself and to nothing else (Motion.SelfOnlyAt, ZeroParadox/Die.lean), and each chart has its own occupancy theorem (ZeroParadox/Settheory/QuineDichotomy.lean, ZeroParadox/Settheory/Wall.lean, ZeroParadox/Settheory/Coalgebra.lean): in a lattice with AbstractSelfApp, or with the membership host an AFAStructure builds, the only occupant is ⊥ (selfApp_quine_role_iff, quineAtomHost_quine_role_iff); in Mathlib's ZFSet, which carries Foundation, the role has no occupant (zfset_quine_role_unoccupied); in the final coalgebra of the identity functor it is occupied (cofix_quine_role_occupied), a one-point chart (cofix_subsingleton), where occupancy restates that chart's existence fork and adds nothing beyond it. This is what ZP-E's DA-1 Path 1 invokes informally. ZP-J encodes the role as a typeclass field, so every result that uses it states it as a hypothesis rather than leaving it as narrative motivation.

Supplying the field is not a test a lattice can fail. Every ZPSemilattice carries an AFAStructure: take the constant-bottom self-application trivialSelfApp (abstractSelfApp_always_inhabited, ZeroParadox/Computability/SelfApp.lean § V) and pass it through the instance toAFAStructure, so that selfMem(x) holds exactly when x = ⊥. Carrying the structure therefore says nothing about a particular lattice. The class encodes a ROLE, and T-EXEC identifies that role's occupant. The selfMem field is itself forced in every instance: selfMem_determines_singleton (Section VII.III) gives {x : L | selfMem(x)} = {⊥} for any AFAStructure lattice L. Where a concrete instance carries content, it is in the map selfMem is defined from (a self-application or a scale, Section VIII.IV), never in selfMem.

Remark R-J.0 — CIC Encoding and the AFA Distinction

In the MachinePhase concrete instance (ZP-K), selfMem is defined as selfMem x := x = ⊥. With this definition, bot_self_mem is proved by rfl: ⊥ = ⊥ holds by reflexivity.

This is the CIC-compatible encoding of AFA self-containment. In Lean 4 (based on the Calculus of Inductive Constructions), the set-theoretic statement ⊥ ∈ ⊥ (ZF+AFA asserts Q ∈ Q of the set Q = {Q}, and nothing of ⊥) is not well-typed on a carrier: x : L is a member of a Set L, and no membership relation between two elements of L is in scope, so ⊥ ∈ ⊥ does not elaborate. Mathlib's own model of ZF sets satisfies Foundation (ZFSet.mem_wf: ∈ is well-founded, from which ZFSet.mem_irrefl follows: no set is a member of itself). The encoding selfMem x := x = ⊥ captures the structural role: "self-containing" means "equals the bottom element." The Lean proof compiles by rfl because self-containing is defined as equality with ⊥.

This is a structural analogy, not a set-theoretic derivation from ZF+AFA. The set-theoretic consequence of AFA used here, that the set Q = {Q} exists and is unique (Aczel 1988, Example 1.3), is not present in the Lean proof, and no equation between ⊥ and Q is written. What the typeclass encodes is the structural consequence: there is a unique element playing the Quine role, and ⊥ is that element.

III. Why a Typeclass Field Rather than a Freestanding Axiom

In the stub version of ZeroParadox/Settheory/SetTheoryAFA.lean, the bridge was a freestanding axiom: ax_j1_quine_join_identity. It stated directly that the Quine atom satisfies the join-identity. This compiled, but the purity check showed T-EXEC depending on ax_j1_quine_join_identity — a named axiom floating

outside any typeclass.

A freestanding axiom is a global assertion the kernel accepts without proof, and it appears in the purity report of every theorem that uses it. A typeclass field adds nothing to that report: T-EXEC depends on no axioms. What the field buys is honesty of scope, not content — it is a hypothesis each theorem carries, and (Section III.II) every lattice can supply it trivially.

The distinction: a freestanding axiom says "trust me, this is true," and the purity check records it. A typeclass field says "given this structure," and every theorem is conditional on it. ZP-J uses the second. The set Q = {Q} stays in ZF+AFA; the Lean carries only the Quine-atom role (CC-2) that set plays.

Section IV: Theorem T-EXEC — The Quine Atom is Bottom

Theorem T-EXEC — Executability of Self-Reference

Statement: Let L be a ZP-A semilattice with AFAStructure. If q : L is a Quine atom (selfMem(q) and q is unique among self-containing elements), then q = ⊥.

Lean: ZeroParadox.t_exec — proved in ZeroParadox/Settheory/SetTheoryAFA.lean. Purity: does not depend on any axioms. ✓

I. The Proof

The proof of T-EXEC is immediate from the typeclass fields. It has three steps:

- hq.2 states: every self-containing element of L equals q. (This is the uniqueness half of IsQuineAtom q.)
- AFAStructure.bot_self_mem states: ⊥ is self-containing. (This is the bridge field — the Quine-atom role (CC-2), played in ZF+AFA by the set Q = {Q}.)
- Applying hq.2 to ⊥ using bot_self_mem gives: ⊥ = q. By symmetry: q = ⊥. QED.

The entire proof is one line in Lean 4: (hq.2 bot AFAStructure.bot_self_mem).symm. No appeal to the join operation. No bridge axiom. No DA-2. Just AFA uniqueness applied at ⊥.

Remark R-J.1

The proof does not use da2_bottom_characterization (from ZP-E). That result — "(∀ x, join S x = x) ↔ S = ⊥" — is used in the derived theorem J1 (Section V), but T-EXEC itself uses neither the order nor the join: it uses only bot_self_mem and the uniqueness half of its own hypothesis IsQuineAtom q (not the quine_unique field).

II. IsQuineAtom bot

A corollary of T-EXEC is that ⊥ itself is a Quine atom. This is the converse direction: not only does the Quine atom equal ⊥ (T-EXEC), but ⊥ equals the Quine atom (bot_is_quine_atom).

Proposition — bot_is_quine_atom
In any AFAStructure lattice L: IsQuineAtom(⊥).
Proof: ⊥ is self-containing by bot_self_mem. Any other self-containing x satisfies x = ⊥ by quine_unique(x, ⊥, selfMem(x), bot_self_mem).
Lean: ZeroParadox.bot_is_quine_atom — does not depend on any axioms. ✓

III. The Full Biconditional

Combining T-EXEC and bot_is_quine_atom yields the full biconditional: IsQuineAtom(q) ↔ q = ⊥. Being the Quine atom and being the bottom element are the same property, stated as a self-containment role (the lattice encoding of the ZF+AFA set Q = {Q}) and in order-theoretic language respectively.

Theorems t_exec_iff and t_exec_triple_iff — Full Equivalence
For any q : L in an AFAStructure lattice:
t_exec_iff: IsQuineAtom(q) ↔ q = ⊥.
t_exec_triple_iff: IsQuineAtom(q) ↔ (q = ⊥ ∧ ∀ x : L, join q x = x).
The three conditions are mutually equivalent: Quine atom (the self-containment role, encoding the set-theoretic Q = {Q}), bottom element (order-theoretic), and join-identity element (algebraic). They are three formulations of one structural role.
Lean: ZeroParadox.t_exec_iff, ZeroParadox.t_exec_triple_iff — do not depend on any axioms. ✓

Section V: Derived Results — J1, and CC-1 Restated

I. J1 — QuineJoinIdentity (Derived)

In the initial stub of ZeroParadox/Settheory/SetTheoryAFA.lean, the claim "the Quine atom satisfies the join-identity" was stated as a freestanding axiom (ax_j1_quine_join_identity). ZP-J replaces it with a theorem.

Theorem J1 — QuineJoinIdentity (formerly Axiom AX-J1)
In any AFAStructure lattice L, if q is the Quine atom, then ∀ x : L, join q x = x.
Proof: q = ⊥ by T-EXEC. Then join q x = join ⊥ x = x by A4 (bot_join, ZP-A). ✓
Status: DERIVED THEOREM. Was axiom ax_j1_quine_join_identity in the stub — now proved from T-EXEC + ZP-A A4. No freestanding axiom remains.
Lean: ZeroParadox.j1_quine_join_identity — does not depend on any axioms. ✓

II. CC-1 Restated

cc1_derived proves: if the initial state S_0 of a state sequence is a Quine atom Q, then $S_0 = \bot$. By t_exec_iff the converse holds as well, since $\bot$ is a Quine atom. So "starts at a Quine atom" and "starts at $\bot$" are the same condition, and CC-1's choice of starting point is restated in terms of the Quine-atom role, not eliminated.

Corollary CC-1, Conditional Form — cc1_derived
Let L be an AFAStructure lattice. Let $S : \mathbb{N} \to L$ be a state sequence (ZP-A D3) and $Q : L$ a Quine atom. If $S(0) = Q$, then $S(0) = \bot$.
Proof: $S(0) = Q$ (hypothesis). $Q = \bot$ by T-EXEC. Therefore $S(0) = \bot$.
Starting at Q and starting at $\bot$ are not two choices — with t_exec_iff they are the same condition. The commitment to start there is not removed; it is stated as starting in the Quine-atom role.
Lean: ZeroParadox.cc1_derived — does not depend on any axioms. ✓

III. Uniqueness of the Bottom Role

A further consequence is algebraic uniqueness: in any ZP-A semilattice (without even requiring AFA structure), at most one element can satisfy the join-identity. This is a clean corollary of da2_bottom_characterization (ZP-E).

Theorem — bot_unique
For any ZP-A semilattice L (no AFA required): if x, y : L both satisfy $\forall$ z, join x z = z and $\forall$ z, join y z = z, then x = y.
Proof: da2_bottom_characterization gives $x = \bot$ and $y = \bot$. Therefore x = y.
Lean: ZeroParadox.bot_unique — does not depend on any axioms. ✓

Section VI: Implications — What Was the Commitment?

I. The Remaining Foundation

T-EXEC and its corollaries carry zero freestanding axioms in the Lean 4 purity check. The entire derivation traces to the fields of two typeclasses: ZPSemilattice (from ZP-A) and AFAStructure (new in ZP-J). No standalone axiom statement appears anywhere in the ZeroParadox/Settheory/SetTheoryAFA.lean proof obligations.

The foundational commitments are the typeclass fields themselves. In ZPSemilattice: A1–A4 (the semilattice axioms). In AFAStructure: selfMem (a predicate), quine_unique (AFA uniqueness), and bot_self_mem (the bridge). They are not axioms floating in the ambient theory — they are hypotheses every theorem here carries. They are not facts about any particular lattice: every ZPSemilattice supplies them trivially (Section III.II).

II. Where the Philosophy Lives Now

The philosophical content of the Quine atom (CC-2) is not in the Lean. It lives in ZP-A as CC-2 (Self-Containment of $\bot$), a Forced Metatheoretic Commitment with two halves: that any host of a

self-membered bottom is not well-founded is proved (quineHost_not_wellFounded, zfSet_no_quine_bottom, ZeroParadox/Settheory/QuineHost.lean), and that a unique Quine atom is the right requirement is argued. It lives too in ZP-E's DA-1 Path 1 (one of three informal arguments for instantiation as execution), and in ZF+AFA it is the set Q = {Q}. What ZP-J adds is the ROLE that set plays, stated as the field bot_self_mem, and a proof (T-EXEC) that whatever fills that role is ⊥.

Because every lattice can fill the role trivially, carrying AFAStructure is not evidence about any particular lattice. In ZF+AFA the set Q = {Q} exists and is unique, a consequence of the axiom (Aczel 1988, Example 1.3). No equation between the framework's ⊥ and that set is written: they sit in different charts, and each chart's occupant of the role is a theorem of that chart (Section III.II). Adopting AFA as the set chart's metatheory stays the commitment (CC-2; ZP-E, Remark R-AFA). The role, stated as a hypothesis where any reader can see it, is what the Lean checks.

III. The Formal Chain

The derivation chain entering ZP-J had one remaining gap: CC-1 was a committed starting point, not a derived one. Sections I–VI restate that commitment in terms of the Quine-atom role, where it becomes an equivalence rather than a bare choice. Sections VII–X extend the chain further: they derive AFAStructure's laws from stronger structures, provide concrete models, and prove decoration uniqueness for every quiver, with a graph-side proof for finite graphs, and decoration existence for every quiver, over abstract DecorationUniverses — analogues of the uniqueness and existence clauses of what AFA asserts as an axiom. Every node is either a proved theorem or a typeclass field:

- ZP-A axioms A1–A4: semilattice structure (ZPSemilattice fields).
- ZP-B: 2-adic topology and irreversibility (proved from Mathlib).
- ZP-C: information theory, L-INF, L-RUN (proved axiom-free or from Mathlib).
- ZP-D: state layer, orthogonality (proved from ZP-A and ZP-B).
- ZP-E: T-SNAP, DA-1, DA-2 (T-SNAP and DA-2 proved axiom-free; DA-1 Path 3 not formalized).
- ZP-J: AFAStructure fields (selfMem, quine_unique, bot_self_mem). T-EXEC, J1, and CC-1 in conditional form: proved axiom-free. Sections VII–X (Aczel connection, abstraction chain, concrete instances, decoration uniqueness): all sorry-free. ✓

DA-1 Path 3 (the AIT/Kolmogorov complexity argument in ZP-E) is not formalized in this corpus: not located as of 2026-11-07, searched over ZeroParadox/**/*.lean in three phrasings. “Path 3”: a comment in ZeroParadox/Category/Category.lean and the docstring of da1_paths_unified (ZeroParadox/Computability/Kleene.lean). “incompressib”: ae_incompressible (ZeroParadox/Information/MaxEntropy.lean), a bound on fair-coin tapes for each prefix-free decoder, and prose mentions in Surprisal.lean, Disjunctive.lean and KleeneChoiceFree.lean. “shorter”, “external” or “outside” before “program”, “generator” or “description”, and “shortest program” or “shortest description”: the docstrings of IsKleeneFixedPoint, da1_paths_unified and description_instantiation_gap_closed (Kleene.lean; the last says “No external program is prior to ⊥” is a requirement the framework imposes “rather than derives here”) and prose in Surprisal.lean. None is a proof of the argument. Kolmogorov complexity itself is now in the build: plainK and KPPlain come from the Lake dependency kolmogorov-complexity-lean (A. Milovanov) at commit 7e71116, and ZeroParadox/Information/BrudnoBridge.lean restates three of its entropy-rate theorems and one randomness criterion for this corpus’s trials and MLRandom. In Shen, Uspensky and Vereshchagin,

Kolmogorov Complexity and Algorithmic Randomness (AMS 2017; SUV), brudno_of_MLRandom and brudno_fair_of_MLRandom are the binary cases of Theorem 148 and Problem 238 (p. 229), pointwise forms along Martin-Löf random tapes, and brudno_ae_trials is the binary case of Problem 239 (p. 229), the almost-everywhere Bernoulli case of Brudno's theorem; the name Brudno is this corpus's label, used by neither SUV on those pages nor the dependency. MLRandom_fair_iff_le_KPPlain is SUV Theorem 94(d) (p. 151), the prefix-complexity form of the Levin–Schnorr criterion, which SUV states as Theorem 90 (p. 146) in monotone complexity. The gap is therefore the Path 3 argument itself, not the absence of K. Before that dependency, a definition of K was not located in the Mathlib then pinned (5450b53) as of 2026-11-03, searched for "Kolmogorov complexity", "KolmogorovComplexity", "algorithmic information", "shortest program" and "description length" (no file), and "Kolmogorov" (Kolmogorov spaces and the Kolmogorov quotient, Kolmogorov processes and the Kolmogorov condition, mentions of the Kolmogorov–Chentsov theorem, the Chapman–Kolmogorov equations, the extension theorem, the zero-one law and an author credit; none a definition of Kolmogorov complexity). ZP-J does not affect DA-1's status.

Section VII: The Aczel Connection — What the Encoding Does Not Settle

Formalised in ZeroParadox/Settheory/AczelConn.lean as an extension of the T-EXEC typeclass encoding. The results are choice-free statements about a lattice predicate; they do not reach Aczel's set-theoretic question.

I. Aczel's Use of Dependent Choice

Aczel (Non-Well-Founded Sets, 1988, ch. 6, Theorem 6.5) proves that for a set-continuous operator Φ, $J\Phi = \bigcup\{x \mid x \subseteq \Phi x\}$ is the largest fixed point of Φ, and that every class X with $X \subseteq \Phi X$ is contained in $J\Phi$ (part 2). In the proof of part (2) he uses the axiom of Dependent Choice (DC) to build an infinite sequence of subsets of X, and notes: "I do not know if this use of the axiom of dependent choices was essential."

The step that uses DC passes from an element a of a CLASS X with $X \subseteq \Phi X$ to a SET x with $a \in x \subseteq \Phi x$, which is needed because $J\Phi$ is a union of sets. That step is where Aczel's question lives, and it is a question about a proof in set theory.

II. The J_self Set and Its Structure

In ZP's encoding the lattice-level counterpart is $J_self = \{x : L \mid selfMem(x)\}$, the set of self-containing elements, defined directly by comprehension on the predicate selfMem. No set-continuous operator is defined and J_self is not built as a union of sets, so the step that uses DC in Aczel's proof has no counterpart here. The results about J_self follow immediately from AFAStructure (Lean: ZeroParadox/Settheory/AczelConn.lean § I):

J_self Theorems (ZeroParadox/Settheory/AczelConn.lean § I)
$\bot \in J_self$: the bottom element is self-containing. (AFAStructure.bot_self_mem.)
J_self_eq_bot: every $x \in J_self$ satisfies $x = \bot$. (One step: quine_unique x ⊥ hx bot_self_mem.)

J_self Theorems (ZeroParadox/Settheory/AczelConn.lean § I)
J_self_eq_singleton_bot: J_self = {⊥}. No Classical.choice ([propext, Quot.sound]). ✓
J_self_is_largest: for any set S of self-containing elements, S ⊆ J_self. This holds by the definition of J_self, for any predicate, and uses no uniqueness: it has the shape of Aczel 6.5 part (2) and is not an instance of it. ✓

quine_unique is used once, to identify the only element of J_self as ⊥. That identification corresponds to no clause of Theorem 6.5, and J_self_is_largest does not use it. The absence of Classical.choice is a measured fact about these Lean terms. It does not show that DC is inessential in Aczel's proof, for this or any operator.

III. The Abstract Principle: A Unique Witness Determines Its Set

The identification is not specific to self-membership. It holds for any predicate with a unique witness. ZeroParadox/Settheory/AczelConn.lean makes this explicit:

Theorem singleton_from_unique_witness (ZeroParadox/Settheory/AczelConn.lean § II)
For any type α and predicate P : α → Prop: if w satisfies P(w) and ∀ x, P(x) → x = w, then {x | P(x)} = {w}.
Proof: pure set extensionality. No Classical.choice ([propext, Quot.sound]). ✓
Application: selfMem_determines_singleton — {x : L | selfMem(x)} = {⊥} for any AFAStructure lattice L.

> Scope note: ZP-J does not answer Aczel's question, for the self-membership case or any other. His question concerns a use of DC inside a proof in set theory; the results here are choice-free statements about a lattice predicate and do not reach that proof. This document takes no position on whether that use of DC is essential.

Section VIII: The Abstraction Chain — ValuationStructure → AbstractSelfApp → AFAStructure

Developed in ZeroParadox/Computability/SelfApp.lean and ZeroParadox/Valuation/Scale.lean after T-EXEC was established; the abstraction chain derives AFAStructure's laws from stronger structures: a fixed-point structure and, beneath it, a valuation structure.

I. The Question: Can AFAStructure's Fields Be Derived?

AFAStructure has three fields: selfMem (a predicate), quine_unique (AFA uniqueness), and bot_self_mem (the bridge). These appear as structural prerequisites — typeclass fields rather than freestanding axioms, but still commitments that any instance must supply — two to PROVE, and selfMem to DEFINE, since it is data rather than a law. The abstraction chain asks: can the two laws themselves be derived from other structure?

The answer is yes, in two steps, with selfMem supplied by a definition along the way. First, AbstractSelfApp provides a self-application operation with unique_fp as a law, from which the two AFAStructure LAWS become derived theorems. Then ValuationStructure explains why ⊥ is the unique fixed point, making even unique_fp a theorem rather than a field.

II. AbstractSelfApp — The Minimal Fixed-Point Structure

AbstractSelfApp encodes the abstract pattern common to both the set-theoretic and 2-adic domains: a self-application operation whose unique fixed point is ⊥.

AbstractSelfApp Typeclass (ZeroParadox/Computability/SelfApp.lean § I)
class AbstractSelfApp (L : Type*) [ZPSemilattice L] with:
(1) selfApp : L → L — the self-application operation.
(2) fixed_bot : selfApp(⊥) = ⊥ — ⊥ is a fixed point.
(3) unique_fp : ∀ x : L, selfApp(x) = x → x = ⊥ — ⊥ is the ONLY fixed point.

From the two laws fixed_bot and unique_fp (selfApp is data), the two AFAStructure LAWS become derived theorems, and selfMem is supplied as a definition (selfMemDerived) rather than proved — it is data:

Derived Results from AbstractSelfApp (ZeroParadox/Computability/SelfApp.lean § II–III)
selfMemDerived x := selfApp(x) = x — self-containment as fixed-point property.
derived_bot_self_mem: selfMemDerived(⊥) — from fixed_bot. ✓
derived_quine_unique: any two self-containing elements are equal — each equals ⊥ by unique_fp, so equal to each other. ✓
selfMem_eq_singleton_bot: {x | selfMemDerived(x)} = {⊥} — via singleton_from_unique_witness. No Classical.choice. ✓
instance toAFAStructure: any AbstractSelfApp gives an AFAStructure. The two laws are theorems, not additional commitments; selfMem is a supplied definition. ✓

III. ValuationStructure — Why ⊥ is the Unique Fixed Point

AbstractSelfApp assumes unique_fp. ValuationStructure derives it. The key insight is the 2-adic argument: if scale increases valuation by 1 at every non-⊥ element, then scale(x) = x implies val(x) = val(x) + 1 — impossible for any finite valuation. Only ⊥, which has infinite valuation (val(⊥) = ⊤), can be a fixed point.

ValuationStructure Typeclass (ZeroParadox/Valuation/Scale.lean § I)
class ValuationStructure (L : Type*) [ZPSemilattice L] with:
(1) scale : L → L — the self-application (scaling) operation.
(2) val : L → ℕ∞ — valuation into the extended naturals.
(3) scale_bot : scale(⊥) = ⊥ — ⊥ is a fixed point of scale.
(4) val_bot : val(⊥) = ⊤ — ⊥ has infinite valuation.
(5) val_unique : ∀ x, val(x) = ⊤ → x = ⊥ — only ⊥ has infinite valuation.

ValuationStructure Typeclass (ZeroParadox/Valuation/Scale.lean § I)
(6) val_scale : ∀ x ≠ ⊥, val(scale(x)) = val(x) + 1 — scale strictly increases valuation at non-⊥ elements.

Derived Results from ValuationStructure (ZeroParadox/Valuation/Scale.lean § II–IV)
val_finite_of_ne_bot: x ≠ ⊥ ⇒ val(x) ≠ ⊤ — contrapositive of val_unique. ✓
scale_ne_fixed: x ≠ ⊥ ⇒ scale(x) ≠ x — val(scale(x)) = val(x) + 1 ≠ val(x) since val(x) is finite. ✓
scale_unique_fp: scale(x) = x ⇒ x = ⊥ — from scale_ne_fixed. ✓
instance toAbstractSelfApp: selfApp = scale; fixed_bot and unique_fp are theorems. No additional fields. ✓

IV. The Full Chain and the 2-Adic Parallel

The complete derivation chain is:

> ValuationStructure → AbstractSelfApp → AFAStructure Four valuation axioms → Two fixed-point fields → Three AFA fields At each step, the LAWS of the lower typeclass become derived theorems; the data fields are supplied, not proved.

What this chain does NOT establish, and the Lean says so explicitly. AbstractSelfApp is a STRUCTURE, not a property: it supplies a chosen self-application map, and carrying one says nothing about the carrier, because every ZPSemilattice carries one. The NO-GO gauge in ZeroParadox/Computability/SelfApp.lean §V records the consequence: because toAFAStructure is an instance, the whole AFA layer — IsQuineAtom, bot_is_quine_atom, T-EXEC — comes free from a degenerate map, so the anti-foundation layer is satisfied by a structure in which nothing refers to anything. The chain is therefore a statement about what FOLLOWS from the structure, not evidence that any particular carrier has it non-trivially. Membership is not an argument; a witness that the framework's real instances differ from the degenerate one is. Each arrow is an instance, so it runs from the stronger hypothesis to the weaker: every ZPSemilattice carries AbstractSelfApp and AFAStructure, while a ValuationStructure on a carrier with any element besides ⊥ forces that carrier to be infinite (valuationStructure_forces_infinite, ZeroParadox/Valuation/ScaleBridge.lean § VI). The chain explains the AFA laws; it does not reduce what must be assumed.

At each level of the chain, the 2-adic parallel holds. At the AbstractSelfApp level: multiplication by 2 in $\mathbb{Q}_2$ has 0 as its unique fixed point ($2x = x \Rightarrow x = 0$, by ring arithmetic), and $\{x \in \mathbb{Q}_2 \mid 2x = x\} = \{0\}$ by singleton_from_unique_witness. At the ValuationStructure level: in $\mathbb{Z}_2$, scale = ×2 and val = 2-adic valuation satisfy all four axioms — in particular, $v_2(2x) = v_2(x) + 1$ for $x \neq 0$ (proved from PadicInt.valuation_mul and PadicInt.valuation_p in Mathlib). No ZPSemilattice or ValuationStructure instance on $\mathbb{Z}_2$ is registered, so the parallel is proved as standalone theorems demonstrating the same proof structure closes both cases. That is a fact about what is declared, not what is possible: ZeroParadox/Valuation/Scale.lean § V proves $\mathbb{Z}_2$ admits a ZPSemilattice carrying a ValuationStructure.

Section IX: Concrete Instances — ℕ∞ and OntologicalStates

Developed in ZeroParadox/Settheory/Model.lean and ZeroParadox/Settheory/OntBridge.lean to ground the abstraction chain in concrete types; OntologicalStates takes a shorter path because a two-element carrier admits no ValuationStructure.

I. $\mathbb{N}\infty$ — The Canonical ValuationStructure Instance

$\mathbb{N}\infty$ = WithTop $\mathbb{N}$ (the extended naturals) carries a ZPSemilattice with join = min and bot = $\top$ (the natural maximum). The ZP partial order reverses $\mathbb{N}\infty$'s natural order: $x \leq y$ iff min x y = y iff $x \geq y$. So $\top$ is the ZP-bottom (valuation ∞, unique fixed point) and 0 is the ZP-maximum (fully constrained).

$\mathbb{N}\infty$ Instances (ZeroParadox/Settheory/Model.lean)
instNatInfZPS: ZPSemilattice $\mathbb{N}\infty$ with join = min, bot = $\top$. A1–A3: min is associative, commutative, idempotent. A4: min $\top$ x = x because $\top$ is the maximum. ✓
instNatInfVal: ValuationStructure $\mathbb{N}\infty$ with scale = $(\cdot + 1)$, val = id. scale_bot: $\top + 1 = \top$ (WithTop.top_add). val_bot: id $\top = \top$. val_unique: id x = $\top \Rightarrow$ x = $\top$. val_scale: $x \neq \top \Rightarrow$ id(x + 1) = id(x) + 1 (rfl). ✓

Derived AFA Content on $\mathbb{N}\infty$ (ZeroParadox/Settheory/Model.lean § III)
natInf_scale_unique_fp: $x + 1 = x \Rightarrow x = \top$. (The unique fixed point of $(\cdot + 1)$ in $\mathbb{N}\infty$ is $\top$.) ✓
natInf_selfMem_singleton: $\{x : \mathbb{N}\infty \mid x + 1 = x\} = \{\top\}$. ✓
Via toAbstractSelfApp and toAFAStructure, $\mathbb{N}\infty$ carries a full AFAStructure with the two laws as theorems and selfMem supplied as a definition. ✓

II. OntologicalStates — The Direct AbstractSelfApp Path

OntologicalStates = {null, exist} (ZP-B's two-element state space) cannot follow the ValuationStructure path: a ValuationStructure on a carrier with any element besides $\bot$ forces that carrier to be infinite (valuationStructure_forces_infinite), so a two-element type admits none. Instead it takes the direct path to AbstractSelfApp.

The self-application operation is the constant-to-null function: every element maps to null. null is the unique fixed point because null $\mapsto$ null (fixed_bot) and exist $\mapsto$ null $\neq$ exist (unique_fp holds vacuously for exist). AFA content follows immediately from the AbstractSelfApp instance.

OntologicalStates Instances (ZeroParadox/Settheory/OntBridge.lean)
instOntZPS: ZPSemilattice OntologicalStates with null-identity join and bot = null. All four axioms proved by case analysis. ✓
instOntSelfApp: AbstractSelfApp OntologicalStates with selfApp = constant-to-null. fixed_bot: null $\mapsto$ null = null (rfl). unique_fp: null $\mapsto$ rfl; exist $\mapsto$ absurd hx (by decide). ✓

Derived AFA Content on OntologicalStates (ZeroParadox/Settheory/OntBridge.lean § III)
ont_bot_self_mem: null is self-containing. ✓

Derived AFA Content on OntologicalStates (ZeroParadox/Settheory/OntBridge.lean § III)
ont_quine_unique: any two self-containing elements of OntologicalStates are equal. ✓
ont_selfMem_singleton: {x | selfMemDerived(x)} = {null}. No Classical.choice. ✓

Two concrete instances, two paths through the chain. ℕ∞ takes the full ValuationStructure route. OntologicalStates bypasses ValuationStructure and connects directly to AbstractSelfApp. Both deliver the same conclusion: the unique element satisfying selfMem (read as x = {x}) is the bottom. The architecture is sound because each type takes the path the mathematics allows.

Section X: APG Decoration Uniqueness and Existence

Formalised in ZeroParadox/Settheory/APG.lean as the most recent addition; this section proves decoration uniqueness for every quiver, with a graph-side proof for finite accessible pointed graphs, and decoration existence for every quiver, over abstract DecorationUniverses — analogues of AFA's uniqueness and existence clauses, not those clauses and not AFA itself — using the typeclass chain established in Sections VIII–IX.

I. Accessible Pointed Graphs and Decorations

An Accessible Pointed Graph (APG) is a directed graph (Quiver) with a distinguished root vertex from which every vertex is reachable via directed paths. AFA is an axiom, not a theorem: every graph has a unique decoration — a labelling of vertices by sets satisfying the membership equation d(v) = {d(w) | v → w} (Aczel 1988, ch. 1 p. 6). Its well-founded counterpart, Mostowski's Collapsing Lemma (p. 4), is a result: every well-founded graph has a unique decoration.

ZP-J's uniqueness result is stated for an abstract DecorationUniverse rather than sets. This avoids ZFSet (which satisfies Foundation, making self-loops impossible) and places the result in the typeclass framework established above.

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} = scale x
collect_val_ge : ∀ (S : Set U) (x : U), x ∈ S → val (collect S) ≥ val x + 1
Exactly two laws. They pin the SINGLETON case and a lower bound; they do not require collect to assemble a parent value from its children.
A valid decoration d : V → U satisfies d(v) = collect({d(w) | v → w}) at every vertex v.

II. The Valuation Argument for Cyclic Vertices

The first key results handle cyclic vertices. The argument is a chain of two inequalities, not an iterated equation. Let w be the next vertex on the cycle and p the path from w back to v. Because w is a child of v and d(v) collects over d of v's children, collect_val_ge gives $\mathrm{val}(d(v)) \geq \mathrm{val}(d(w)) + 1$. Following p, the lemma path_val_chain (private to ZeroParadox/Settheory/APG.lean) gives $\mathrm{val}(d(w)) \geq \mathrm{val}(d(v)) + \mathrm{length}(p)$. path_val_chain's own proof recurses along p, accumulating at least one unit of valuation per edge — the bound above is itself reached by that recursion, not asserted outright. Together these force val(d(v)) to exceed itself, which no finite value in $\mathbb{N}\infty$ can do, so $\mathrm{val}(d(v)) = \top$ (infinite) and therefore $d(v) = \bot$. No assumption is made about how many children a vertex has.

The iterated form $d(v) = \mathrm{scale}^k(d(v))$ is not how this proof proceeds. The class laws relate collect to scale only through collect_singleton, so composing decoration equations yields that form when each vertex's set of child decorations on the cycle is a singleton — for example, when each vertex has exactly one child. The scale^k lemmas below are genuine results about periodic points of scale, and kCycle_node_eq_bot takes that equation as a hypothesis rather than deriving it.

Cyclic Vertex Theorems (ZeroParadox/Settheory/APG.lean §§ III–VII')
val_iterate: $\mathrm{val}(\mathrm{scale}^k(x)) = \mathrm{val}(x) + k$ for $x \neq \bot$. ✓
scale_iterate_unique_fp: $\mathrm{scale}^k(x) = x \Rightarrow x = \bot$ for $k \geq 1$. ✓
pureSelfLoop_decoration_eq_bot: any valid decoration assigns $\bot$ to a pure self-loop vertex. ✓
kCycle_node_eq_bot: if $d(v) = \mathrm{scale}^k(d(v))$ for some $k \geq 1$ under any valid decoration, then $d(v) = \bot$. The cycle equation is a hypothesis, not derived. ✓
cyclic_decoration_eq_bot: any vertex with a directed cycle through itself receives $\bot$ under any valid decoration. Proof route: collect_val_ge and path_val_chain, with val_finite_of_ne_bot; it uses neither val_iterate nor scale_iterate_unique_fp. ✓

III. The Acyclic Case and the "Infinite Period"

Cyclic vertices have a finite period k — the path returns in k steps, and a valuation bound around the cycle forces $\bot$. Acyclic vertices have an "infinite period" — the path never returns to them, so they carry no cycle of their own for that bound to run around. Read as a coincidence of two charts, the finite k of a cycle is the zero side and its absence is the infinity side. The proved half holds of one vertex in both charts at once: a cyclic vertex receives the order-theoretic zero $\bot$, and $\bot$'s valuation is $\top$, infinite depth (cyclic_decoration_eq_bot with val_bot). The implication runs from the cycle to that value only; no value is claimed for an acyclic vertex from its period alone. The pairing is stated for DecorationUniverses: in AFA set theory a graph with a cycle can picture a set other than Q, such as $0^* = \{\varnothing, 0^*\}$ (Aczel 1988, Example 1.5).

The hardness of the acyclic proof is that "infinite period" gives you nothing to induct on. Reach cardinality — the cardinality of $\{w \mid \mathrm{Path}\ v\ w\}$, the set of vertices reachable from v — serves as a finite proxy. For any child w of an acyclic vertex v, the reach of w is a strict subset of the reach of v (since v cannot be reached from w without creating a cycle). Reach cardinality is therefore strictly decreasing, providing a well-founded measure for the induction.

That measure is the graph side, and it is where finiteness is spent. The value side needs none: decoration_unique_of_val (ZeroParadox/Settheory/APG.lean § IX) inducts on the valuation of the decorating value instead. A value of finite valuation m is collected from children whose values have valuation below m (collect_val_ge), and a value of valuation $\top$ is $\bot$ of U (val_unique). It holds on every quiver, finite or not. Credit: this adapts a result Aczel sets as an exercise, that the Foundation axiom implies AFA2 (every graph has at most one decoration) (Aczel, Non-Well-Founded Sets, CSLI 1988, Exercise 2.1, p. 20; AFA1 and AFA2 are defined on p. 19), and 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 that exercise is a point of infinite depth, $\bot$ of U, admitted through val_unique. The two sides are related one way: a cyclic vertex's value has valuation $\top$ (cyclic_decoration_eq_bot with val_bot); no converse is claimed.

Acyclic Induction, Global Uniqueness and Existence (ZeroParadox/Settheory/APG.lean §§ VIII–X)
acyclic_induction_step: if two valid decorations d_1, d_2 agree on all children of a vertex v, they agree on v. (The two child image sets are equal by Set.ext, then rewrite — ordinary congruence, consuming no class axiom.) ✓
decoration_unique_of_val, the main uniqueness result: for any quiver V, finite or not, with no APG argument, any two valid decorations d_1, d_2 into any DecorationUniverse are equal. Proof: strong induction on a finite valuation; valuation $\top$ is $\bot$ of U by val_unique. decoration_unique_finite restates decoration_unique with [Finite V] in place of [Fintype V], as a corollary. Both [propext, Quot.sound]. ✓
decoration_unique [Fintype V]: for any finite APG with root r and any two valid decorations d_1, d_2 into any DecorationUniverse, $d_1 = d_2$. Proof: strong induction on $\lvert\{w \mid \text{Path } u\ w\}\rvert$; cyclic case by cyclic_decoration_eq_bot; acyclic case by acyclic_induction_step with IH on children. ✓
exists_decoration and existsUnique_decoration (§ X): every quiver has exactly one valid decoration into every DecorationUniverse. Recursion on the accessible part, $\bot$ of U elsewhere, with the case split on the valuation, not on accessibility. [propext, Quot.sound]. ✓

decoration_unique_of_val is the ZP analogue of the uniqueness clause of AFA (an axiom, not a theorem): for any quiver, finite or not, at most one valid decoration exists into any DecorationUniverse. Its proof consumes collect_val_ge and val_unique and no finiteness. exists_decoration is the analogue of the existence clause: at least one exists, so existsUnique_decoration gives exactly one. decoration_unique is the finite, graph-side proof, for finite APGs; it consumes collect_val_ge (never collect_singleton), val_unique through val_finite_of_ne_bot, and [Fintype V] for the acyclic descent, and its statement is a special case of decoration_unique_of_val. decoration_unique_of_val and decoration_unique each give a sufficient condition for decoration uniqueness, not a characterisation. None of these constructs a specific AFA model or derives AFA's axioms from ZP's: a decoration into an abstract DecorationUniverse is not a model of AFA.

Traceability Register — ZP-J

Claim	Grounded In	New assumption?	Status
T-EXEC: Quine atom = ⊥	AFAStructure.bot_self_mem + the uniqueness half of IsQuineAtom q	None — typeclass fields	Lean: t_exec — axiom-free ✓
J1: Quine join-identity	T-EXEC + ZP-A A4 (bot_join)	None	Lean: j1_quine_join_identity — axiom-free ✓ (was axiom ax_j1 in stub)
CC-1, conditional form: $S_0 = Q \Rightarrow S_0 = \bot$	T-EXEC	None	Lean: cc1_derived — axiom-free ✓ (restates ZP-A's conditional claim; converse by t_exec_iff)
bot_is_quine_atom	AFAStructure.bot_self_mem + quine_unique	None	Lean: bot_is_quine_atom — axiom-free ✓
t_exec_iff: IsQuineAtom ↔ = ⊥	T-EXEC + bot_is_quine_atom	None	Lean: t_exec_iff — axiom-free ✓
bot_unique: join-identity is unique	da2_bottom_characterization	None	Lean: bot_unique — axiom-free ✓ (no AFA required)
J_self = {⊥} (no Classical.choice)	AFAStructure.quine_unique — one step	None	Lean: J_self_eq_singleton_bot — no Classical.choice ✓
singleton_from_unique_witness	Set extensionality (propext)	None	Lean: singleton_from_unique_witness — [propext, Quot.sound]; no Classical.choice ✓
AbstractSelfApp → AFAStructure	AbstractSelfApp.fixed_bot + unique_fp	None	Lean: toAFAStructure — selfMem := selfMemDerived (a definition); bot_self_mem and quine_unique are theorems ✓
ValuationStructure → AbstractSelfApp	ValuationStructure.val_scale + val_unique	None	Lean: scale_unique_fp, toAbstractSelfApp — [propext, Classical.choice, Quot.sound] ✓
ℕ∞ : ValuationStructure	ℕ∞ arithmetic (WithTop.top_add)	None	Lean: instNatInfZPS, instNatInfVal — each [propext, Classical.choice, Quot.sound] ✓
OntologicalStates : AbstractSelfApp	Constant-to-null map; two-element case analysis	None	Lean: instOntSelfApp — axiom-free ✓
cyclic_decoration_eq_bot	collect_val_ge + path_val_chain + val_finite_of_ne_bot	None	Lean: cyclic_decoration_eq_bot — [propext, Classical.choice, Quot.sound] ✓

Claim	Grounded In	New assumption?	Status
decoration_unique [Fintype V]	cyclic_decoration_eq_bot + acyclic_induction_step + Set.ncard	[Fintype V] — finite graph assumption	Lean: decoration_unique — propext, Classical.choice, Quot.sound ✓
decoration_unique_of_val (any quiver)	collect_val_ge + val_unique + acyclic_induction_step	None — no finiteness, no APG	Lean: decoration_unique_of_val — [propext, Quot.sound]; no Classical.choice ✓
decoration_unique_finite [Finite V]	decoration_unique_of_val	[Finite V] — in the statement only; the proof does not use it	Lean: decoration_unique_finite — [propext, Quot.sound]; no Classical.choice ✓
exists_decoration, existsUnique_decoration (any quiver)	accDec (Acc.rec) + collect_val_ge + val_unique + decoration_unique_of_val	None — no finiteness, no APG	Lean: exists_decoration, existsUnique_decoration — [propext, Quot.sound]; no Classical.choice ✓
AFAStructure.bot_self_mem	The Quine-atom role (CC-2), played in ZF+AFA by the set Q = {Q}	Typeclass field — a hypothesis; every ZPSemilattice meets it trivially	Not a theorem; a hypothesis T-EXEC carries.

Open Items Register — ZP-J

Item	Status	Description
CC-1 (ZP-A) derivability	RESTATED — not forced	cc1_derived: a state sequence that starts at a Quine atom starts at ⊥, and by t_exec_iff the converse holds, so the two starting conditions are equivalent. No freestanding axiom; the choice of starting point remains, stated through the role.
AX-J1 bridge axiom	CLOSED — J1 derived	The stub version had ax_j1 as a freestanding axiom. The final version derives J1 from T-EXEC + ZP-A A4. Axiom eliminated.
AFAStructure concrete instances	CLOSED — multiple instances	MachinePhase (ZP-K): machinePhaseAFA : AFAStructure MachinePhase. ℕ∞: instNatInfZPS + instNatInfVal → AbstractSelfApp → AFA content. OntologicalStates: instOntSelfApp → AFA content directly.
Aczel DC question	NOT ADDRESSED	J_self = {⊥} is proved with no Classical.choice, and J_self_is_largest holds by definition. Neither is a statement about Aczel's proof of Theorem 6.5 in set theory, so this document takes no position on whether his use of DC is essential.

Item	Status	Description
DA-1 Path 1 formalisation	PARTIAL — APG case proved	DA-1 Path 1 (ZP-E) invokes the Quine atom (CC-2) informally. ZP-J T-EXEC formalises the role's occupant in the lattice chart; no equation with the set Q = {Q} is stated. decoration_unique_of_val (§ X) proves uniqueness for every quiver over abstract DecorationUniverses, existsUnique_decoration (§ X) existence and uniqueness for every quiver, and decoration_unique uniqueness for finite APGs. The full ZF+AFA set-theoretic bridge — showing the ZP types literally satisfy the ZF+AFA axioms — remains outside Lean scope.
ZP-A CC-1 cross-check	RESOLVED	ZP-A's CC-1 box now matches this document: a Conditional Claim at ZP-A scope, which cc1_derived and t_exec_iff restate as an equivalence of starting conditions, not a forcing of the starting point.
Formal ZPSemilattice instance for a ValuationStructure type	CLOSED — registered on $\mathbb{N}\infty$	ZeroParadox/Settheory/Model.lean registers both instances on $\mathbb{N}\infty$. ZeroParadox/Valuation/Scale.lean § V proves $\mathbb{Z}_2$ admits a ZPSemilattice carrying a ValuationStructure, as an existence statement, without registering an instance; no registered instance on $\mathbb{Z}_2$ was located in ZeroParadox/**/*.lean on 2026-09-13, re-checked 2026-11-06.

End of ZP-J | Theorem T-EXEC: Executability of Self-Reference | CC-1 restated as an equivalence — no freestanding axioms | J_self = {⊥} with no Classical.choice | Abstraction chain: ValuationStructure → AbstractSelfApp → AFAStructure | Instances: $\mathbb{N}\infty$, OntologicalStates | decoration_unique_of_val: any two valid decorations of any quiver, finite or not, into a DecorationUniverse agree | decoration_unique: the same for a finite APG, by a graph-side proof | existsUnique_decoration: every quiver has exactly one valid decoration into a DecorationUniverse | All results sorry-free in Lean 4