THE ZERO PARADOX

ZP-J: Executability of Self-Reference

Version 2.0 | May 2026

v2.0: Sections VII–X added — Aczel DC-free connection, abstraction chain, concrete instances, APG decoration uniqueness. | v1.2: Minor wording fix. | v1.1: Remark R-J.0 — Lean encoding scope. | v1.0: T-EXEC; all ZPJ.lean theorems axiom-free.

This document establishes Theorem T-EXEC (Executability of Self-Reference): in any ZP-A join-semilattice with AFA (Anti-Foundation Axiom) grounding, the unique Quine atom Q — the element satisfying $Q = \{Q\}$ — is provably the bottom element $\bot$. This closes the bridge between the set-theoretic and order-theoretic layers of the framework. CC-1 from ZP-A, which stated "$S_0 = \bot$" as a modelling commitment, becomes a derived theorem.

Version 2.0 extends the core T-EXEC result in four directions: it establishes that Aczel's use of Dependent Choice is unnecessary for the self-membership case (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 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 $\bot$, then $\bot \leq S(n)$ for all n. This follows trivially from T2 ($\bot$ is the global minimum), so its formal content is essentially: we are choosing $\bot$ 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 $\bot$? The answer is yes, provided the lattice has AFA grounding.

II. The Implicit Identification

ZP-E's DA-1 Path 1 already states: $\bot = \{\bot\}$ (Quine atom, ZF+AFA). This identification — the bottom element is the unique self-containing set under AFA — 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 (ZP-J T-EXEC): it is forced — by the AFA identification $\bot = \{\bot\}$, encoded structurally in the AFAStructure typeclass.

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: $x \in x$ is impossible in ZF. This rules out self-referential sets by fiat.

The Anti-Foundation Axiom (AFA, Aczel 1988) replaces Foundation with a universal existence and uniqueness result: every accessible pointed graph (APG) has a unique set-theoretic decoration. Under AFA, self-containing sets are not only possible but uniquely characterised. 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 self-containing elements are equal. This means the self-membership property uniquely identifies an element — a fingerprint that belongs to exactly one object in the universe.

AFA Uniqueness (AFAStructure.quine_unique)
For any type L with AFA structure: if $x, y \in L$ both satisfy selfMem(x) and selfMem(y), then $x = y$.
Informally: the Quine atom is unique. Self-containment is a property held by at most one element of any AFA-structured type.
Lean: AFAStructure.quine_unique — encoded as a typeclass field, not a theorem, because AFA is a foundational axiom and cannot be derived from type-theoretic principles alone. It is a structural prerequisite for any AFA-grounded lattice.

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 ZPJ.lean captures exactly this. Its three fields are sufficient to derive T-EXEC with no additional axioms. The full AFA machinery (APGs, bisimulation, decoration) provides the informal justification for why any concrete lattice grounded in ZF+AFA satisfies these three fields — but the formal derivation requires only the fields themselves.

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 (ZPJ.lean § I)
class AFAStructure (L : Type*) [ZPSemilattice L] with:
(1) selfMem : L → Prop — the self-membership predicate. selfMem(x) means x contains itself as a member in the AFA sense.
(2) quine_unique : ∀ x y : L, selfMem(x) → selfMem(y) → x = y — AFA uniqueness. At most one element of L is self-containing.
(3) bot_self_mem : selfMem(⊥) — the bridge field. The bottom element of the lattice is self-containing. This is the formal encoding of ⊥ = {⊥}.

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 set-theoretic terms: ⊥ ∈ ⊥, i.e. ⊥ = {⊥}. In the framework: the null state contains itself as its sole content. This is not a new claim — it is the identification that ZP-E's DA-1 Path 1 already invokes informally. ZP-J encodes it as a typeclass field, making it a verifiable structural prerequisite rather than a narrative motivation.

Any concrete lattice L that claims to be AFA-grounded must prove bot_self_mem as part of its AFAStructure instance. If it cannot, it is not genuinely AFA-grounded — the identification ⊥ = {⊥} is part of what "AFA-grounded" means.

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.

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

This is the CIC-compatible encoding of AFA self-containment. In Lean 4 (based on the Calculus of Inductive Constructions), the set-theoretic statement $\bot \in \bot$ — meaning $\bot$ literally contains itself as a member under ZF+AFA — is not directly formalizable. CIC lacks the membership relation $\in$ of set theory. The encoding selfMem x := x = $\bot$ 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 $\bot$.

This is a structural analogy, not a set-theoretic derivation from ZF+AFA. The full set-theoretic content of AFA — that $\bot$ literally contains itself as a member of the AFA universe — is not present in the Lean proof. What the typeclass encodes is the structural consequence: there is a unique element playing the Quine role, and $\bot$ is that element.

III. Why a Typeclass Field Rather than a Freestanding Axiom

In the stub version of ZPJ.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.

Encoding the commitment as a typeclass field is philosophically and formally cleaner. A freestanding axiom is a global assertion that the proof checker accepts without verification. A typeclass field is a requirement: any instance of AFAStructure must supply a proof of bot_self_mem. The commitment is not asserted once and forgotten — it is checked at every instantiation. Concrete lattices must earn their AFA status.

The distinction: a freestanding axiom says "trust me, this is true." A typeclass field says "prove it for your specific lattice, or it does not compile." ZP-J uses the second. The philosophical commitment ($\bot = \{\bot\}$) becomes a proof obligation at every concrete instantiation.

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 = $\bot$.

Lean: ZeroParadox.ZPJ.t_exec — proved in ZPJ.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: $\bot$ is self-containing. (This is the bridge field — $\bot = \{\bot\}$ encoded structurally.)
- Applying hq.2 to $\bot$ using bot_self_mem gives: $\bot = q$. By symmetry: $q = \bot$. 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 is purely order-theoretic: it uses only quine_unique and bot_self_mem.

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.ZPJ.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 in set-theoretic and order-theoretic language respectively.

Theorem t_exec_iff — Full Equivalence

For any q : L in an AFAStructure lattice:

IsQuineAtom(q) ↔ q = ⊥ ↔ ∀ x : L, join q x = x.

The three conditions are mutually equivalent: Quine atom (set-theoretic), bottom element (order-theoretic), and join-identity element (algebraic). They are three formulations of one structural role.

Lean: ZeroParadox.ZPJ.t_exec_iff — does not depend on any axioms. ✓

Section V: Derived Results — J1 and CC-1 as Theorems

I. J1 — QuineJoinIdentity (Derived)

In the initial stub of ZPJ.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 $\forall x : L$, 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.ZPJ.j1_quine_join_identity — does not depend on any axioms. ✓

II. CC-1 Derived

ZP-A CC-1 is now a theorem. If the initial state S_0 of a state sequence is the Quine atom Q, then $S_0 = \bot$ — a consequence of T-EXEC, not a modelling choice.

Theorem CC-1 (Derived) — 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) = ⊥.
Proof: S(0) = Q (hypothesis). Q = ⊥ by T-EXEC. Therefore S(0) = ⊥.
The modelling commitment "we choose ⊥ as the initial state" is replaced by "the Quine atom IS ⊥, structurally." Starting at Q and starting at ⊥ are not two choices — they are the same state, identified by structure.
Lean: ZeroParadox.ZPJ.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 = ⊥ and y = ⊥. Therefore x = y.
Lean: ZeroParadox.ZPJ.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 ZPJ.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). These are the structural prerequisites for any lattice that claims ZP-A + AFA grounding. They are not axioms floating in the ambient theory — they are proof obligations that any concrete instance must discharge.

II. Where the Philosophy Lives Now

The philosophical content of $\bot = \{\bot\}$ has not disappeared. It has moved. Previously it lived in a narrative comment in ZP-A (CC-1's informal motivation) and in ZP-E's DA-1 Path 1 (one of three informal arguments for instantiation as execution). Now it lives in the definition of AFAStructure itself — specifically in bot_self_mem.

This is the correct location for a foundational claim. It is not hidden in a proof; it is stated in the typeclass definition where any reader can see it. Any lattice that wants to use ZP-J's results must provide a proof that its bottom element is self-containing. If it cannot, it is simply not an AFA-grounded lattice in the sense required by this framework.

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 close that gap. Sections VII–X extend the chain further: they reduce the axiom load of AFAStructure, provide concrete models, and prove the full AFA decoration uniqueness theorem for finite graphs. 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 outside Lean scope).
- ZP-J: AFAStructure fields (selfMem, quine_unique, bot_self_mem). T-EXEC, J1, CC-1: proved axiom-free. v2.0: 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) remains outside Lean scope for the same reason it always has: Kolmogorov complexity is uncomputable and absent from Mathlib. ZP-J does not affect DA-1's status.

Section VII: The Aczel Connection — DC-Free Results

Formalised in ZPJ_AczelConn.lean as an immediate extension of the T-EXEC typeclass encoding; the DC-free observation follows directly from quine_unique.

I. Aczel's Use of Dependent Choice

Aczel (Non-Well-Founded Sets, 1988, ch. 6) proves that $J\Phi = \bigcup\{x \mid x \subseteq \Phi x\}$ is the largest pre-fixed-point of a set-continuous operator Φ. The proof uses the axiom of Dependent Choice (DC) to construct an ω-chain, and Aczel explicitly notes: "I do not know if this use of the axiom of dependent choices was essential."

DC is used when you must build a witness step by step because the fixed point cannot be identified structurally. In ZF+AFA, the Quine atom is unique by the axiom itself. Once uniqueness is available, construction is unnecessary — identification suffices.

II. The J_self Set and Its Structure

In ZP's encoding, the analogue of Aczel's $J\sum$ for the self-membership operator is $J\_self = \{x : L \mid selfMem(x)\}$ — the set of self-containing elements. The key results about J_self follow immediately from AFAStructure (Lean: ZPJ_AczelConn.lean § I):

J_self Theorems (ZPJ_AczelConn.lean § I)
bot ∈ J_self: the bottom element is self-containing. (AFAStructure.bot_self_mem.)
J_self_eq_bot: every x ∈ J_self satisfies x = ⊥. (One step: quine_unique x ⊥ hx bot_self_mem.)
J_self_eq_singleton_bot: J_self = {⊥}. Proved without DC. ✓
J_self_is_largest: for any set S of self-containing elements, S ⊆ J_self. (Aczel 6.5 part (2), self-membership case — without DC.) ✓

Where Aczel's proof requires an ω-chain to arrive at $J\sum$, the ZP proof is one line: quine_unique applied once identifies the unique self-containing element as ⊥. The chain is redundant because the fixed point is forced, not constructed.

III. The Abstract Principle: Uniqueness Eliminates DC

The argument is not specific to self-membership. It holds for any predicate with a unique witness. ZPJ_AczelConn.lean makes this explicit:

Theorem singleton_from_unique_witness (ZPJ_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 DC, no Classical.choice. ✓
Application: selfMem_determines_singleton — {x : L | selfMem(x)} = {⊥} for any AFAStructure lattice L.

Scope note: the DC-free result applies to the self-membership operator because AFA provides uniqueness (quine_unique) directly. Whether DC can be eliminated for general set-continuous operators Φ depends on whether uniqueness holds for each Φ-fixed-point — an open question, as Aczel's "I do not know" reflects. ZP does not claim DC-freedom for the general case.

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

---

Developed in ZPJ_SelfApp.lean and ZPJ_Scale.lean after T-EXEC was established; the abstraction chain reduces the axiom load of AFAStructure by deriving its fields from a minimal fixed-point 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 discharge. The abstraction chain asks: can these three commitments themselves be derived from something more primitive?

The answer is yes, in two steps. First, AbstractSelfApp provides a self-application operation and proves unique_fp, from which all three AFAStructure fields 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 (ZPJ_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 these two fields, all three AFAStructure fields become derived theorems:

Derived Results from AbstractSelfApp (ZPJ_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. DC-free. ✓
instance toAFAStructure: any AbstractSelfApp gives an AFAStructure. All three fields are theorems, not additional commitments. ✓

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 (ZPJ_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.
(6) val_scale : ∀ x ≠ ⊥, val(scale(x)) = val(x) + 1 — scale strictly increases valuation at non-⊥ elements.

Derived Results from ValuationStructure (ZPJ_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 fields of the lower typeclass become derived theorems.

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). $\mathbb{Z}_2$ cannot be a formal instance because it is a ring, not a ZPSemilattice; the parallel is proved as standalone theorems demonstrating the same proof structure closes both cases.

Section IX: Concrete Instances — ℕ∞ and OntologicalStates

Developed in ZPJ_Model.lean and ZPJ_OntBridge.lean to ground the abstraction chain in concrete types; OntologicalStates takes a shorter path because its two-element structure cannot support val_scale.

I. ℕ∞ — The Canonical ValuationStructure Instance

ℕ∞ = WithTop ℕ (the extended naturals) carries a ZPSemilattice with join = min and bot = ⊤ (the natural maximum). The ZP partial order reverses ℕ∞'s natural order: $x \leq_{5?} y$ iff min x y = y iff x ≥ y. So ⊤ is the ZP-bottom (valuation ∞, unique fixed point) and 0 is the ZP-maximum (fully constrained).

ℕ∞ Instances (ZPJ_Model.lean)
instNatInfZPS: ZPSemilattice ℕ∞ with join = min, bot = ⊤. A1–A3: min is associative, commutative, idempotent. A4: min ⊤ x = x because ⊤ is the maximum. ✓
instNatInfVal: ValuationStructure ℕ∞ with scale = (· + 1), val = id. scale_bot: ⊤ + 1 = ⊤ (WithTop.top_add). val_bot: id ⊤ = ⊤. val_unique: id x = ⊤ ⇒ x = ⊤. val_scale: x ≠ ⊤ ⇒ id(x + 1) = id(x) + 1 (rfl). ✓

Derived AFA Content on ℕ∞ (ZPJ_Model.lean § III)
natInf_scale_unique_fp: x + 1 = x ⇒ x = ⊤. (The unique fixed point of (· + 1) in ℕ∞ is ⊤.) ✓
natInf_selfMem_singleton: {x : ℕ∞ | x + 1 = x} = {⊤}. ✓
Via toAbstractSelfApp and toAFAStructure, ℕ∞ carries a full AFAStructure with all three fields as theorems. ✓

II. OntologicalStates — The Direct AbstractSelfApp Path

OntologicalStates = {null, exist} (ZP-B's two-element state space) cannot follow the ValuationStructure path: a finite two-element type has no room for val_scale (val(scale x) = val(x) + 1 would require infinitely many distinct values). 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 ↦ null (fixed_bot) and exist ↦ null ≠ exist (unique_fp holds vacuously for exist). AFA content follows immediately from the AbstractSelfApp instance.

OntologicalStates Instances (ZPJ_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 ↦ null = null (rfl). unique_fp: null ↦ rfl; exist ↦ absurd hx (by decide). ✓

Derived AFA Content on OntologicalStates (ZPJ_OntBridge.lean § III)
ont_bot_self_mem: null is self-containing. ✓
ont_quine_unique: any two self-containing elements of OntologicalStates are equal. ✓
ont_selfMem_singleton: {x | selfMemDerived(x)} = {null}. DC-free. ✓

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 self-containing element is the bottom. The architecture is sound because each type takes the path the mathematics allows.

Section X: APG Decoration Uniqueness

Formalised in ZPJ_APG.lean as the most recent addition; this section proves the full AFA decoration uniqueness theorem for finite accessible pointed graphs, 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's central theorem states that every APG has a unique set-theoretic decoration — a labelling of vertices by sets satisfying the membership equation $d(v) = \{d(w) \mid v \to w\}$.

ZP-J's version of this theorem 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.

DecorationUniverse Typeclass (ZPJ_APG.lean § II)
class DecorationUniverse (U : Type*) [ZPSemilattice U] [ValuationStructure U] with:
(1) collect_singleton : collect {x} = scale(x) — a singleton set decorates to scale of its element.
(2) collect_ext : collect s_1 = collect s_2 when $s_1 = s_2$ — collect respects set equality.
(3) collect_val_ge : val(collect s) ≥ val(x) + 1 for x ∈ s and x ≠ ⊥ — valuation of a collected set is strictly bounded below.
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. If a vertex v lies on a directed cycle of length k, then composing the decoration equation around the cycle gives d(v) = scale□(d(v)). The valuation argument then forces d(v) = ⊥: if d(v) ≠ ⊥, then val(scale□(d(v))) = val(d(v)) + k ≠ val(d(v)), contradicting d(v) = scale□(d(v)).

Cyclic Vertex Theorems (ZPJ_APG.lean §§ III–VII')
val_iterate: val(scale□(x)) = val(x) + k for x ≠ ⊥. ✓
scale_iterate_unique_fp: scale□(x) = x ⇒ x = ⊥ for k ≥ 1. ✓

Cyclic Vertex Theorems (ZPJ_APG.lean §§ III–VII')
pureSelfLoop_decoration_eq_bot: any valid decoration assigns ⊥ to a pure self-loop vertex. ✓
kCycle_node_eq_bot: if d(v) = scale□(d(v)) under any valid decoration, then d(v) = ⊥. ✓
cyclic_decoration_eq_bot: any vertex with a directed cycle through itself receives ⊥ under any valid decoration. ✓

III. The Acyclic Case and the "Infinite Period"

Cyclic vertices have a finite period k — the path returns in k steps, and the valuation argument closes the equation. Acyclic vertices have an "infinite period" — the path never returns, so there is no cycle length k to close the equation. This is the same zero-versus-infinity identification that runs throughout the framework: the finite k of a cycle is the zero side; its absence is the infinity side.

The hardness of the acyclic proof is that "infinite period" gives you nothing to induct on. Reach cardinality — the cardinality of {w | 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.

Acyclic Induction and Global Uniqueness (ZPJ_APG.lean §§ VIII–IX)
acyclic_induction_step: if two valid decorations d_1, d_2 agree on all children of an acyclic vertex v, they agree on v. (From collect_ext applied to the decoration equations.) ✓
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 |{w | Path u w}|; cyclic case by cyclic_decoration_eq_bot; acyclic case by acyclic_induction_step with IH on children. ✓

> decoration_unique is the ZP version of AFA's decoration uniqueness theorem: for any finite APG, at most one valid decoration exists into any DecorationUniverse. The proof uses only the three typeclass axioms and ValuationStructure — it characterises when decoration uniqueness holds. It does not construct a specific AFA model or derive AFA's axioms from ZP's.

Traceability Register — ZP-J v2.0

Claim	Grounded In	New assumption?	Status
T-EXEC: Quine atom = ⊥	AFAStructure.quine_unique + AFAStructure.bot_self_mem	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)

Claim	Grounded In	New assumption?	Status
CC-1 (Derived): $S_0 = Q \Rightarrow S_0 = \bot$	T-EXEC	None	Lean: cc1_derived — axiom-free ✓ (was ZP-A conditional claim)
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	ZPE.da2_bottom_characterization	None	Lean: bot_unique — axiom-free ✓ (no AFA required)
J_self = {⊥} (DC-free)	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 — axiom-free ✓
AbstractSelfApp → AFAStructure	AbstractSelfApp.fixed_bot + unique_fp	None	Lean: toAFAStructure instance — all fields as theorems ✓
ValuationStructure → AbstractSelfApp	ValuationStructure.val_scale + val_unique	None	Lean: scale_unique_fp, toAbstractSelfApp — axiom-free ✓
ℕ∞ : ValuationStructure	ℕ∞ arithmetic (WithTop.top_add)	None	Lean: instNatInfZPS, instNatInfVal — axiom-free ✓
OntologicalStates : AbstractSelfApp	Constant-to-null map; two-element case analysis	None	Lean: instOntSelfApp — axiom-free ✓
cyclic_decoration_eq_bot	val_iterate + scale_iterate_unique_fp	None	Lean: cyclic_decoration_eq_bot — axiom-free ✓
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 ✓
AFAStructure.bot_self_mem	AFA (ZF+AFA) — ⊥ = {⊥}	Typeclass field — proof obligation at instantiation	Not a theorem; a structural prerequisite.

Open Items Register — ZP-J v2.0

Item	Status	Description
CC-1 (ZP-A) derivability	CLOSED — T-EXEC (ZP-J v1.0)	CC-1 is now a theorem. The Quine atom = ⊥ is structurally derived. No freestanding axiom. No modelling commitment beyond AFAStructure typeclass fields.

Item	Status	Description
AX-J1 bridge axiom	CLOSED — J1 derived (v1.0)	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∞ (v2.0): instNatInfZPS + instNatInfVal → AbstractSelfApp → AFA content. OntologicalStates (v2.0): instOntSelfApp → AFA content directly.
Aczel DC question (self-membership)	CLOSED — DC-free (v2.0)	For the self-membership operator, DC is unnecessary. J_self = {⊥} proved in one step via quine_unique. General set-continuous operators: still open (Aczel's "I do not know" stands).
DA-1 Path 1 formalisation	PARTIAL — APG case proved (v2.0)	DA-1 Path 1 (ZP-E) invokes ⊥ = {⊥} informally. ZP-J T-EXEC formalises the identification. decoration_unique (§ X) proves uniqueness for finite APGs over abstract DecorationUniverses. 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 label update	OPEN — editorial	ZP-A still labels CC-1 as a Conditional Claim. With T-EXEC established, the label can be updated to Derived Theorem (citing ZP-J T-EXEC). Mathematical content unchanged; editorial update only.
Formal ZPSemilattice instance for a ValuationStructure type	OPEN — gap in chain	ZPJ_Scale.lean proves the ValuationStructure → AbstractSelfApp chain abstractly. Connecting a concrete type (e.g. a 2-adic type or ZP state space) to both ZPSemilattice and ValuationStructure in the same instance requires a bridge file importing ZP-A and ZP-B. The ℕ∞ instance (ZPJ_Model.lean) fills this for the abstract model; a concrete ZP-grounded instance remains future work.

End of ZP-J v2.0 | Theorem T-EXEC: Executability of Self-Reference | CC-1 derived — no freestanding axioms | DC-free: J_self = {⊥} without Dependent Choice | Abstraction chain: ValuationStructure → AbstractSelfApp → AFAStructure | Instances: ℕ∞, OntologicalStates | decoration_unique: any two valid decorations of a finite APG agree | All v2.0 results sorry-free in Lean 4