Guarded Recursion for Meta-Learning Factories

Finite construction records, behavioral semantics, and executable replay

Abstract. An autonomous factory that can construct a successor has two recursive descriptions. Its finite construction records are inductive, while observations of its controller unfold coinductively. For an arity-bounded polynomial in an ∞-topos, we construct the homotopy-initial algebra of construction records and a unique guarded solution whose limit gives terminal behavior. A truncation theorem relates finite records with holes to controller observations. A conserved credit budget bounds the number of paid births and yields a sufficient observation horizon for the executable fragment. The example learns a binary initialization for a support/query adaptation rule and passes it to a daughter. Lean checks the discrete semantics and both unary and branching accounting. In QEMU, daughters booted from complete ELF images emitted by their parents and matched the Lean reference over 188 valid boots, including 110 verified births. Eight malformed manifests were rejected.

Keywords. Guarded recursion, ∞-topos, polynomial functor, coalgebra, meta-learning, simulation.

1 Problem and scope

A factory controller can deliver a product or install a successor with the same factory interface. A product with sensing, decision, and actuation, such as a vehicle with a driving controller, motivates the distinction between these roles. For example, Tesla's Model 3 manual describes Full Self-Driving (Supervised) as a driver-assistance feature that requires an attentive driver [9]. Here, autonomy refers to a simulated controller making its own decision to produce a successor inside a run.

A finite witness of successful assembly admits structural induction. A controller that can start a successor admits finite observations even when the specification has no given maximum depth. Its observable behavior includes when work stops, the costs paid, and the state delivered to a successor. A meta-learner makes that state depend on adaptation across tasks.

Self-replicating lunar manufacturing was investigated as a design study [7], and physical modular self-reproduction has been demonstrated in a different setting [8]. The experiments here concern a virtual factory with three simulated components and a self-copying controller image. The factory constructs a complete daughter executable in a guest. A host harness starts a fresh guest from the emitted bytes. This operational definition allows a reproducible test of inherited state and recursive execution.

The paper gives an ∞-categorical interpretation, an accounting invariant for branching lineages, and a discrete adequacy theorem checked by Lean. The formalization is aligned with the deterministic guest by an independent executable comparison.

2 Construction and observation

2.1 Ambient category

Let $\mathcal{X}$ be an ∞-topos with initial object 0 and terminal object 1. One choice is $\mathrm{Fun}(\mathcal{C}^{op}, \mathcal{S})$, where a small category $\mathcal{C}$ indexes observation or interface contexts and $\mathcal{S}$ denotes spaces. For parameters varying in $P \in \mathcal{X}$, the same constructions live in the slice $\mathcal{X}_{/P}$. In the latter case 1 denotes the identity on P, and products are taken over P. An object of labels may therefore carry both parameter dependence and automorphisms. We use standard properties of ∞-topoi, including left exactness of filtered colimits and pointwise computation of limits in functor categories [1].

Definition 2.1 (Constructor signature)**.** Fix $d < \infty$ and label objects $A_k \in \mathcal{X}$ for $0 \le k \le d$. Define the endofunctor

$$F(X) = \coprod_{k=0}^{d} A_k \times X^k. \tag{1}$$

An element of A_k describes an event with k successor slots. The nullary labels record completed or terminal events. In the controller, event labels can record observations, the choice of initialization, and debit information. In construction syntax, they can record assembly operations and component interfaces.

The signature in (1) uses ordered slots. One can give A_k an action of the symmetric group Σ_k and take homotopy orbits of the diagonal action on $A_k \times X^k$. The sequential-colimit proof below also applies to this variant because homotopy orbits are colimits. Polynomial functors and their compositional structure have a broader theory [2].

An F-algebra is (Z, a) with $a : FZ \to Z$. The space of algebra maps from (W, w) to (Z, a) is the homotopy equalizer of the two maps

$$\mathrm{Map}(W, Z) \rightrightarrows \mathrm{Map}(FW, Z),$$

given by $f \mapsto fw$ and $f \mapsto aF(f)$. Its points consist of a map and a specified commutative-square homotopy. Initiality means that this entire mapping space is contractible [3]. An F-coalgebra is (S, γ) with $\gamma : S \to FS$. An execution state contains every value on which the next event depends. Dependence on hidden state or previously seen observations belongs in S.

2.2 Finite construction records

Let $W_0 = 0$, $W_{n+1} = FW_n$, and take the initial chain $i_0 : 0 \to F0$, $i_{n+1} = Fi_n$. There is a canonical algebra structure on the colimit of this chain.

Theorem 2.2 (Homotopy-initial construction algebra)**.** *For the signature* (1)*,*

$$W = \operatorname*{colim}_{n\in\mathbb{N}} W_n \quad \textit{has} \quad w : FW \xrightarrow{\simeq} W.$$

The algebra (W, w) *is initial in* $\mathrm{Alg}_F(\mathcal{X})$*.*

Proof. Sequential colimits commute with finite products in an ∞-topos. Product with a constant label preserves colimits, as does coproduct over the arities. Thus F preserves the initial-chain colimit. Cofinality of the successor shift gives $FW \simeq \mathrm{colim}_n W_{n+1} \simeq W$.

Fix an algebra (Z, a) and set $M_n = \mathrm{Map}(W_n, Z)$. Write $r_n : M_{n+1} \to M_n$ for restriction and $t_n : M_n \to M_{n+1}$ for $x \mapsto aF(x)$. For $n \geq 1$, naturality gives $r_n t_n \simeq t_{n-1} r_{n-1}$. The endomorphisms $T_0 = \mathrm{id}$ and $T_{n+1} = t_n r_n$ give a coherent map of the tower. The identification $M_0 \simeq 1$ supplies the initial compatibility. On $\lim M_n \simeq \mathrm{Map}(W, Z)$ this map sends f to $aF(f)w^{-1}$, as can be checked on the chain inclusions. Algebra maps form $\mathrm{Eq}_h(\mathrm{id}, \lim T_n)$, which is $\lim_n \mathrm{Eq}_h(\mathrm{id}, T_n)$ because limits commute.

For $u : B \to A$ and $v : A \to B$, applying u or v to a point and its specified path induces maps $\mathrm{Eq}_h(\mathrm{id}, vu) \rightleftarrows \mathrm{Eq}_h(\mathrm{id}, uv)$. The composites apply the relevant endomorphism to the point and path. They are homotopic to the identity by the specified path, whose naturality square commutes up to the canonical path-composition homotopy. Apply this with $u = r_n$, $v = t_n$ and use $r_n t_n \simeq T_n$. Each bonding map on the equalizers is an equivalence and the zeroth equalizer is contractible. The limit is contractible. □

If all A_k are discrete, W is the set of finite rooted terms whose node labels and arities agree. Its structure map installs a root above the supplied subterms. An algebra map evaluates the term into any algebra of summaries, such as event logs or accumulated costs. A label may contain additional local evidence of feasibility. Without such evidence, initiality gives the syntax and its fold, while validity is a predicate on that syntax.

2.3 Higher symmetries

Consider $\mathcal{X} = \mathcal{S}$, $A_0 = BG$, $A_2 = 1$, and $A_1 = 0$. Here BG is the groupoid of a component interface whose automorphism group is G. For an ordered full binary tree shape T with $\ell(T)$ leaves, the corresponding summand of W is

$$\prod_{j=1}^{\ell(T)} BG.$$

This follows by induction on the finite chain and passage to the colimit, since the inclusions in this example add tree shapes. For $G = \mathbb{Z}/2$, the automorphism group over T is $(\mathbb{Z}/2)^{\ell(T)}$. Replacing BG by $K(G, m)$ for abelian G and $m \geq 2$ gives summands with nontrivial π_m. Taking connected components retains the tree shape and identifies these distinct higher data. Thus the denotation can carry higher structure even when the branching shape is deterministic.

3 Guarded controller semantics

3.1 The topos of observation depths

Define

$$\mathcal{T}(\mathcal{X}) = \mathrm{Fun}(\mathbb{N}^{op}, \mathcal{X}).$$

An object is a tower $X_0 \leftarrow X_1 \leftarrow \cdots$. This diagram category is an ∞-topos. A left exact localization presenting $\mathcal{X}$ induces a pointwise left exact localization presenting $\mathcal{T}(\mathcal{X})$. For presheaf $\mathcal{X}$, it is the presheaf category on $\mathcal{C} \times \mathbb{N}$. The delay functor satisfies

$$(\triangleright X)_0 = 1, \qquad (\triangleright X)_{n+1} = X_n.$$

The first restriction is the terminal map and later restrictions come from X. This is right Kan extension along the successor inclusion of $\mathbb{N}^{op}$. There is a natural $X \to \triangleright X$, given by restriction at positive indices. This is the observation-depth interpretation of the later modality from the topos of trees [4].

Let F also denote pointwise application to a tower. Starting from $B_0 = 1$, put $B_{n+1} = FB_n$. For $p_n : B_{n+1} \to B_n$, let p_0 be the terminal map and set $p_{n+1} = Fp_n$. A state with $n + 1$ observations available can emit its current event and give n observations to each successor.

Theorem 3.1 (Guarded unfolding)**.** *The tower* $Y_n = B_{n+1}$ *has an equivalence* $Y \simeq F(\triangleright Y)$*. The space of towers equipped with such a specified equivalence, with structure-preserving equivalences as morphisms, is contractible. For a coalgebra* $\gamma : S \to FS$*, the maps*

$$o_0 = ! : S \to 1, \qquad o_{n+1} = F(o_n)\gamma \tag{2}$$

form a coherent cone to the B_n*. The cone with these unfolding equations is unique up to contractible choice.*

Proof. The displayed solution satisfies the equations on objects and restrictions. At stage 0 a solution consists of an object with a chosen equivalence to $F1$. That space is contractible. Over a solution through stage n, stage $n + 1$ adds an object with a chosen equivalence to $F(Y_n)$. The naturality square for the new restriction identifies that arrow, under the chosen equivalences, with the restriction prescribed by $F\triangleright$. The space of arrows equipped with a path to a specified arrow is contractible. These data determine the next finite tower. The spine of a linearly ordered simplex determines the functor up to contractible coherent extension. The restrictions of solution spaces therefore have contractible fibers. Taking the limit proves the first assertion.

For the cone, $p_0 o_1 \simeq o_0$ by terminality. Inductively, $p_{n+1} o_{n+2} = F(p_n o_{n+1})\gamma \simeq F(o_n)\gamma = o_{n+1}$. Recursively applying F to the cone on the previous finite tower and precomposing with γ supplies the higher compatibilities. Equivalently, on the mapping-space tower $M_n = \mathrm{Map}(S, B_n)$ the cone is a fixed point of the endomorphism given at stage zero by the terminal map and at stage $n+1$ by $x \mapsto F(p_n x)\gamma$. Here $p_n x$ means postcomposition. The endomorphism factors at each stage through the preceding restriction. Its homotopy equalizer with the identity is contractible by the same restriction argument as in Theorem 2.2. □

Terminal coalgebra. For the ordered signature (1), F preserves inverse sequence limits. To see this, put $A = \coprod_k A_k$ and view $F(X) \to A$ in the slice. Under $\mathcal{X}_{/A} \simeq \prod_k \mathcal{X}_{/A_k}$ its components are $A_k \times X^k \to A_k$. Each component preserves limits because powers and pullback of $X \to 1$ preserve limits. The underlying object of an inverse sequence limit in the slice is its limit in $\mathcal{X}$, since the indexing category has contractible nerve and the constant A diagram has limit A. Thus $F(\lim B_n) \simeq \lim FB_n$, and the shift gives a coalgebra on $B = \lim B_n$. It is terminal by the dual initial-chain argument of Theorem 2.2, and its map from S is the cone of Theorem 3.1. In sets, $F(X) = O \times (1 + X)$ gives nonempty finite traces with a stop marker and infinite traces [5].

Change of context. For a geometric inverse-image functor g^* between ∞-topoi, transporting the label objects transports the polynomial. Finite products and all colimits are preserved. Consequently g^* commutes with the initial-chain colimit and each finite observation stage, including the restrictions. This applies in particular to pullback between slices. Commuting with an external limit $\lim B_n$ additionally requires preservation of that limit. At higher types an identification of full observations consists of a coherent family of identifications in the tower.

3.2 Truncated construction records

Let $W^\sharp$ be the initial algebra of $G(X) = 1 + F(X)$ given by Theorem 2.2. The added nullary constructor hole marks a boundary where the observation has stopped unfolding. A genuine terminal event is still a labeled node of arity zero. Let $\mathsf{node} : FW^\sharp \to W^\sharp$ be the other part of the structure map. Define

$$j_0 = \mathsf{hole} : 1 \to W^\sharp,$$
$$j_{n+1} = \mathsf{node} \circ F(j_n) : B_{n+1} \to W^\sharp.$$

The cut T_k replaces subrecords at depth k by hole. Formally, T_0 is constant at hole, and the decomposition $W^\sharp \simeq 1 + FW^\sharp$ gives

$$T_{k+1} = [\,\mathsf{hole},\ \mathsf{node} \circ F(T_k)\,].$$

In this equation the root has depth zero.

Theorem 3.2 (Compatibility of construction and observation)**.** *Let $p_{n,k} : B_n \to B_k$ be the restriction for $k \le n$. There are canonical homotopies*

$$T_k j_n \simeq j_k p_{n,k}, \qquad T_k j_n o_n \simeq j_k o_k.$$

In particular, the cut records $j_n o_n$ recover the observation tower with its depth restrictions.

Proof. At $k = 0$ both composites are the constant hole map. For $k+1 \le n+1$, use $p_{n+1,k+1} = F(p_{n,k})$. The left composite equals $\mathsf{node} F(T_k j_n)$, which is homotopic to $\mathsf{node} F(j_k p_{n,k}) = j_{k+1} p_{n+1,k+1}$ by induction. The homotopies respect composition of restrictions by the functorial construction. The second formula follows by precomposing with o_n and using the cone identities. For recovery at fixed depth, j_0 is a coproduct inclusion and therefore a monomorphism. By induction $F(j_n)$ is a monomorphism because finite products and disjoint coproducts preserve monomorphisms in an ∞-topos. The node map is also a coproduct inclusion. Thus j_n identifies B_n with the subobject of records it represents. □

4 Meta-learning and conservation

4.1 A two-level learner

A task episode $e = (D_s, D_q)$ supplies support data and query data. A hyperparameter $h \in H$ selects an initialization. The inner map $U_h(D_s)$ adapts this initialization to a task. The outer objective for a nonempty finite episode multiset E is

$$J_E(h) = \sum_{(D_s, D_q) \in E} \ell\big(U_h(D_s), D_q\big). \tag{3}$$

A deterministic selector M minimizes (3) over the chosen finite H and retains the inherited h when it is among the minimizers. For the two-element H used here, that rule resolves every tie. This is a finite support/query objective of the form used for learning an initialization for rapid adaptation [6]. The child inherits the selected hyperparameter and adapts afresh to its own episodes.

Composition in context. Let D be the object of generation states other than the initialization, and write $S = D \times H$. Given a morphism $m : D \times H \to H$ interpreting the outer update and a feasibility-and-assembly map $a : S \to FS$, the round map is

$$S \xrightarrow{(\pi_D, m)} S \xrightarrow{a} FS.$$

One can retain the input initialization in the event label by carrying it as an additional parameter through a. In the slice over a task context, equivariant data and update maps give the same composition there. The unfolding is natural in coherent coalgebra maps: if $\gamma' f \simeq F(f)\gamma$, then $o'_n f \simeq o_n$ coherently in n. Indeed, both families have the same zero stage and satisfy the unfolding equations (2). Thus a change of controller representation that preserves

the update and assembly coalgebra preserves its observations.

For the guest, $H = \{0, 1\}$. A support observation s is either a bit or absent, and a query q is a bit. The inner update is $U_h(s) = s$ if present and $U_h(\varnothing) = h$ otherwise. Loss is $[U_h(s) \neq q]$. Let $t(g) = [g \bmod 4 < 2]$ be the target at generation g and $\bar{t} = 1 - t$. The training episode list E_g and validation episode list V_g are

$$\begin{aligned} E_{2j} &= [(\varnothing, t), (\varnothing, t), (\varnothing, \bar{t}), (\bar{t}, \bar{t})], \\ E_{2j+1} &= [(\varnothing, t), (\varnothing, \bar{t}), (t, t), (\bar{t}, \bar{t})], \\ V_g &= [(\varnothing, t), (\bar{t}, \bar{t}), (\varnothing, t), (t, t)]. \end{aligned}$$

In each line $t = t(g)$ for that line's generation. On even generations $M = t$ with training loss 1, against 2 for $\bar{t}$. On odd generations both choices have training loss 1. The validation loss $Q_g(h)$ is 0 when $h = t(g)$ and 2 otherwise. The rule adapts to the support observation on each episode, and the missing-support episodes make the initialization observable. Validation uses a separate deterministic episode list from the same task generator.

4.2 State, transition, and inheritance

The concrete state is $s = (g, b, h)$, with generation g, natural credit balance b, and inherited bit h. Set $\hat{h} = M(E_g, h)$, $q = Q_g(\hat{h})$, and $c = 3 + q$. The controller emits $o(s)$, the round record containing s, the two training losses, $\hat{h}$, the validation losses for the selected and fixed priors, c, and the optional child. Its coalgebra for $F(X) = O \times (1 + X)$ is

$$\gamma(s) = \begin{cases} (o(s), \mathsf{next}(g + 1, b - c, \hat{h})) & b \geq c, \\ (o(s), \mathsf{stop}) & b < c. \end{cases} \tag{4}$$

The three base credits correspond to the three component flags. The extra charge ties learner performance to feasibility in the model. The task generator depends on g and the transition is deterministic. A lineage may differ in length solely because of inheritance. At $(g, b) = (1, 3)$, inherited 1 leads to a daughter with $(g, b, h) = (2, 0, 1)$. Resetting the bit to 0 gives cost 5, so it stops at the same input generation. The odd-generation tie makes this comparison sensitive specifically to the inherited bit.

4.3 Accounting for branching lineages

The unary guest implements a special case of the following invariant. A funded state carries $b \in \mathbb{N}$. An expansion replaces it by $k \leq d$ funded child states with balances b_i, emits a cost $c \geq \delta > 0$, and satisfies

$$b = c + \sum_{i=1}^{k} b_i. \tag{5}$$

A paid expansion has at least one child. An unexpanded state, including a state that stops, retains its unspent balance in the frontier accounting. Any parent that participates again must itself be represented by a funded successor. Thus the invariant counts all outstanding claims, including any continuation of the parent.

Proposition 4.1 (Conservation and finite horizon)**.** *After any finite sequence of paid expansions from a root with balance b_0,*

$$\sum_{u \in \text{frontier}} b_u + \sum_{v \in \text{expanded}} c_v = b_0. \tag{6}$$

There are at most $\lfloor b_0/\delta \rfloor$ paid expansions, at most $1 + d\lfloor b_0/\delta \rfloor$ created state nodes, and at most $\lfloor b_0/\delta \rfloor + 1$ observation levels along a lineage. If every transition either stops or performs a paid expansion, the cut records of Theorem 3.2 stabilize by that level.

Proof. Initially the sole balance is b_0. For each expansion, replace the removed balance b by the debit and child balances in (5); this preserves (6). After m paid events the debit sum is at least δm, and all residual balances are nonnegative. The node count uses at most d new nodes per expansion. An observed path consists of paid expansions and at most one final stop. This bounds its required observation depth. □

The selector may alter the costs and therefore change realized lineages. Accounting uses the conservation equation and the lower bound on each debit. If the controller can wait without consuming credits, the count still bounds paid births. A bound on clock steps requires a bound on waiting.

5 Executable adequacy

5.1 Inductive records and finite observations

For a deterministic coalgebra $\gamma : S \to O \times (1 + S)$ in sets, define $\mathrm{obs}_0(s) = [\,]$ and

$$\mathrm{obs}_{n+1}(s) = \begin{cases} [o] & \gamma(s) = (o, \mathsf{stop}), \\ o :: \mathrm{obs}_n(s') & \gamma(s) = (o, \mathsf{next} s'). \end{cases} \tag{7}$$

This is the list decoding of (2). At the truncation boundary the decoder forgets the successor tag unless the event label records it. The concrete round label has an explicit optional child field, so the reported logs retain the tag at every emitted round.

Let R be the free algebra of $G_{\mathrm{list}}(X) = 1 + O \times X$. Write end for the nullary constructor and $\mathrm{extend}(o, r)$ for the unary constructor. Define $R_n(s)$ by (7) with $[\,]$ replaced by end and cons by extend. In the stop case the result is $\mathrm{extend}(o, \mathrm{end})$. Equivalently, the fold from $W^\sharp$ to R sends a hole to end and both kinds of labeled nodes to extend, with an empty tail for a nullary labeled node. The universal map $\mathrm{fold} : R \to \mathrm{List}(O)$ uses $[\,]$ and cons as the target constructors.

Theorem 5.1 (Finite-depth adequacy)**.** *For any deterministic unary coalgebra in sets,*

$$\mathrm{fold}(R_n(s)) = \mathrm{obs}_n(s).$$

For $k \leq n$, taking the first k observations of $\mathrm{obs}_n(s)$ gives $\mathrm{obs}_k(s)$. The compatible family satisfying (7) is unique. For (4), $\mathrm{obs}_n(s)$ stabilizes for $n \geq \lfloor b/3 \rfloor + 1$.

Proof. For depth zero the fold returns the empty list. At positive depth both definitions prepend the same o. A stopping branch has no tail, and a continuing branch has the same tail by induction on depth. Prefix compatibility follows by induction on k with n and s generalized. When $k > 0$, also $n > 0$, and the two traces have the same head. For a stop there is nothing to truncate after the head. For a successor use the induction hypothesis with $k-1$ and $n-1$. Uniqueness follows by induction on the stage equations. Finally, each successor decreases the balance by at least three and the final stop contributes one observation. Equivalently, induction on k proves that $b < 3k$ implies $\mathrm{obs}_{k+j}(s) = \mathrm{obs}_k(s)$ for all j. □

5.2 Lean proof boundary

`Factory/Semantics.lean` contains the abstract unary observation function, the finite-record fold, and the compatible-family object. The Lean kernel checks uniqueness of the record fold, finite-depth adequacy, and uniqueness of the compatible family. It also checks optimality of the finite selector, credit conservation along the lineage, and stabilization of the observation for every natural input balance and every extension of the stated horizon. `example_lineage` and `example_inheritance` are concrete decision proofs.

The artifact uses the Lean 4.19.0 standard library [10]. The transcript lists the axioms of the exported results with `#print axioms`. The reported dependencies are drawn from Lean's standard `propext`, `Classical.choice`, and `Quot.sound`. The executable computes the finite reference trace. `Factory/Branching.lean` checks the cut identity, injectivity of the stage embeddings, and the accounting bound for branching ledgers in sets. The ∞-topos theorems are proved in Sections 2 and 3. The C guest and boot path are checked by differential replay on the specified inputs.

6 QEMU experiments

6.1 Boot and handoff protocol

Each 32-bit Multiboot guest runs one factory generation with a single virtual CPU under QEMU's TCG accelerator [11]. Each lineage begins with a freestanding ELF seed image and a Multiboot module for its starting state. The VM has no network interface or writable guest disk. The seed manifest is a 24-byte little-endian record of six unsigned 32-bit words: magic, format version, generation, credits, inherited bit, and component mask. The expected mask is 7, the union of the simulated frame, controller, and fabricator flags. The guest validates the incoming manifest, computes one controller round, assembles the simulated component flags if it can pay, and emits a JSON observation over serial. For a birth it copies its mapped ELF prefix, including program headers, and replaces the embedded manifest. The emitted image carries the controller and the complete daughter state. The guest sends the image and the manifest as hex bytes, then QEMU exits through the ISA debug-exit device.

The Python replay harness records the serial output, checks the embedded manifest against the Lean prediction, and verifies that the executable matches the seed's normalized loadable bytes outside the manifest slot. It passes the unmodified daughter image to QEMU as the next kernel. The daughter boots from its embedded state without an external manifest. Each observed child must appear as a subsequent boot. The host provides the emulator and schedules each guest. The C interface accepts $b \le 120$, $h \in \{0,1\}$, and $g + \lfloor b/3 \rfloor \le 10^6$. These bounds keep guest arithmetic in range and the largest input can produce at most 40 births. The Lean theorems use unbounded natural numbers.

6.2 Design and measured results

The suite comprises six named scenarios, a grid of 72 initial states, and eight malformed inputs. The grid covers all four generation residues, credit balances 0 through 8, and both priors. The named cases include the high-credit and high-generation limits, the unaffordable case, and the matched inheritance intervention. For every valid boot the harness compares the complete round JSON to the compiled Lean reference and checks daughter bytes when present. Malformed inputs test missing components, version, magic, short or long records, out-of-range prior, credits, and generation.

Input (g,b,h)	Boots	Births	Query loss
Lineage $(0,16,0)$	6	5	0
Inherited $(1,3,1)$	2	1	0
Reset $(1,3,0)$	1	0	2
Low credit $(0,2,0)$	1	0	0
Max credit $(0,120,0)$	41	40	0
Max generation $(10^6,2,1)$	1	0	0
Grid (72 inputs)	136	64	—
All valid runs	188	110	—

Table 1: QEMU TCG results. Query loss sums validation errors over observed rounds, including the terminal round.

All 188 valid boots matched the Lean reference, 110 produced verified daughters, and the eight malformed inputs returned the expected rejection. The six-round lineage had balances $16, 13, 10, 7, 4, 1$ and selected priors $1, 1, 0, 0, 1, 1$. Its aggregate query error was 0. Evaluating the fixed priors 0 and 1 on the same six generations gave 8 and 4 errors respectively. These are counterfactual query scores on a shared set of tasks. Enforcing those priors during execution can change which generations are reached. The matched inherited/reset case at $(1,3)$ compares actual births with the same generation, budget, and episode generator.

The periodic target and chosen starting state make correct inheritance persist through ties in the main lineage. Starting on an odd generation with a mismatched bit gives

the counterexample in Table 1. The results establish the behavior of the finite objective and its effect on the declared budget in this generator. Generalization to other task distributions requires separate evidence.

6.3 Replication protocol

The accompanying bundle contains the LaTeX source, Lean sources, the guest sources and linker script, the replay harness, the proof transcript, tool versions, all observed serial logs, manifests and daughter images, and a machine-readable expected summary. On a host with Lean 4.19.0, GNU make, a compiler and binutils supporting i386 ELF, Python 3, and QEMU with the specified PC machine model, `make verify` builds and checks the Lean code, builds the guest, and repeats the differential comparison. It checks the resulting summary against the captured one. `make paper` builds the two-column manuscript. The README gives the commands and output locations. The reported replay used QEMU 10.0.13 with the versioned `pc-i440fx-7.2` machine model, GCC 14.2.0, and Lean 4.19.0.

7 Interpretation

The initial algebra gives a universal evaluation rule for finite construction witnesses, including labels with higher symmetries. The guarded coalgebra gives a universal rule for observations of successive execution stages. The cut theorem connects them at each finite depth. With positive conserved credits, the cut captures the entire run at a calculable depth. The executable intervention shows that inherited meta-state can determine the existence of the next factory even when assembly code and starting resources are identical.

The production model assumes a fixed component signature, an exogenous task generator, and a cost rule. A model of physical closure would need an operational account of component sourcing, metrology, repair, and the infrastructure that starts and verifies a successor. The QEMU evidence covers the emulated controller, assembly flags, and byte-for-byte reproduction of the bootable factory described above.

Artifact and assistance statement. The manuscript, proof sources, and experiment were prepared with AI assistance. The reproducibility bundle contains the complete computational inputs and reported outputs.

References

[1] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009.

[2] N. Gambino and J. Kock. Polynomial functors and polynomial monads. *Math. Proc. Cambridge Philos. Soc.*, 154(1), 153–192, 2013.

[3] S. Awodey, N. Gambino, and K. Sojakova. Homotopy-initial algebras in type theory. arXiv:1504.05531, 2015.

[4] L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory. *Log. Methods Comput. Sci.*, 8(4), 2012.

[5] B. Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge University Press, 2016.

[6] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *Proc. ICML*, PMLR 70, 1126–1135, 2017.

[7] NASA/ASEE study team. Replicating systems concepts: self-replicating lunar factory and demonstration. *Advanced Automation for Space Missions*, NASA CP-2255, 1982.

[8] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. Self-reproducing machines. *Nature*, 435, 163–164, 2005.

[9] Tesla. *Model 3 Owner's Manual*, U.S. edition. Full Self-Driving (Supervised) and limitations.

[10] Lean FRO and contributors. Lean 4.19.0 release notes, 2025.

[11] QEMU contributors. QEMU system invocation manual.