Certified Mealy Machines as Sweedler Comodels

An explicit comparison with dependent polynomial verification

Research note for the categorical systems tutorial

12 October 2026

Abstract

Let p be a set-valued polynomial interface and let r be a proof-relevant specification in the sense of Aberlé: precondition witnesses $C(a)$ and response-dependent post-condition witnesses $D(a, c, b)$. Making explicit Aberlé's arrow-category observation, we construct a polynomial endofunctor $\widehat{P}$ of the arrow topos $\mathbf{Arr}(\mathbf{Set})$ lying over p, and identify the proof fibres of its free monad with the dependent free programs FreeDep. We prove that state-passing runners from $\widehat{T} = \widehat{P}^*$ to the identity monad correspond, naturally in the state arrow, to coalgebras of an explicit endofunctor $\widehat{F}$ on $\mathbf{Arr}(\mathbf{Set})$. These coalgebras are ordinary Mealy machines equipped with a family of certified states and witness-preserving transitions. Their final coalgebra is the arrow $\sum_m \mathrm{DepMealy}(p, r, m) \to \mathrm{Mealy}(p)$. Verified implementations give monad morphisms of the certified free theories; restriction of runner coalgebras along these morphisms executes the underlying program and threads the evidence, agreeing with Aberlé's `run-mealyD`/`prog→mealyD`. All results are set-level and require neither a general dependent Sweedler double category nor a claim about higher proof types. The principal contribution is the category-valued runner comparison, not the arrow-polynomial representation already noted by Aberlé.

Contents

1 The comparison problem **2**

2 From dependent specifications to an arrow polynomial **2**

2.1 The free certified programs 3

3 Arrow state passing and a typed Yoneda calculation **4**

4 Measuring comonad and the final dependent Mealy object **6**

5 Verified implementations as tight interpretations **7**

6 Invariants, genuinely proof-relevant certificates, and equations **8**

6.1 The invariant state interface 8

6.2 A genuinely proof-relevant example: sums of two squares 9

6.3 What changes when the state laws are imposed? 10

7 World-typed machines and precise scope **10**

8 The category-valued comparison and related work **11**

9 Conclusion **13**

1 The comparison problem

An ordinary Mealy machine for a polynomial interface $p = (A, B)$ responds to a command $a \in A$ with a result $b \in B(a)$ and a new state. Aberlé's dependent Mealy construction [1, §§5–6] additionally records evidence: when a precondition $c \in C(a)$ is supplied, the machine produces a witness in $D(a, c, b)$ and a certified successor. The operational responses remain those of the underlying machine.

The question addressed here is precise: can we realise *both* the machines and their verified-program action as comodels and tight theory interpretations in a Sweedler runner setting? We answer yes in the arrow category of sets. As Aberlé observes in [1, §5.1], specifications may be represented by polynomial functors on $\mathbf{Type}^{\to}$ over the ordinary interface; our set-level version below makes this observation explicit. Our new comparison is the runner/cofree-comodel and implementation theorem, not the observation that specifications are arrow polynomials. The input is an ordinary interface with a proof-relevant specification, not yet an independently world-changing operational interface; that extension is indicated in §7.

We use the runner notation of McDermott–Rivas–Uustalu [2]: for monads T, R on a cartesian closed category $\mathcal{E}$, a runner with residual state S is a monad map

$$\alpha : T \longrightarrow \mathrm{St}_S(R), \qquad \mathrm{St}_S(R)(X) = \big(R(S \times X)\big)^S.$$

Here $\mathrm{St}_S(R)$ is the conjugate monad along $S \times - \dashv (-)^S$; no independently supplied strength on R is required. For a *free* monad P^*, such runners are equivalent to natural transformations $P \to \mathrm{St}_S(R)$. We use this elementary characterisation, not a prior existence theorem for every abstract Sweedler measuring hom.

2 From dependent specifications to an arrow polynomial

Fix a small polynomial interface

$$p(X) = P(X) = \coprod_{a \in A} X^{B(a)}$$

and witness families

$$C(a) \quad (a \in A), \qquad D(a, c, b) \quad (a \in A,\ c \in C(a),\ b \in B(a)).$$

No proof-irrelevance or uniqueness of witnesses is imposed. Following Aberlé's observation in [1, §5.1], we spell out the polynomial induced on the set-valued arrow category. An arrow $u : Y \to X$ represents a family $Y_x = u^{-1}\{x\}$ of witnesses over $x \in X$. Put $\mathcal{A} = \mathbf{Arr}(\mathbf{Set}) = \mathbf{Set}^{(0 \to 1)}$.

Definition 2.1 (Certified polynomial)**.** For $u : Y \to X$, define $\widehat{P}(u)$ to be the arrow

$$\coprod_{a\in A}\ \coprod_{c\in C(a)}\ \coprod_{k:B(a)\to X}\ \prod_{b\in B(a)}\ \prod_{d\in D(a,c,b)} Y_{k(b)} \longrightarrow \coprod_{a\in A} X^{B(a)}, \tag{1}$$

which forgets c and the witness-valued continuations, retaining (a, k). Morphisms of arrows act by mapping the ordinary continuations and their witnesses. Thus $\operatorname{cod} \widehat{P} = P \operatorname{cod}$ on the nose, up to harmless choices of coproduct association.

For $a \in A$ and $c \in C(a)$, let

$$J_{a,c} : \coprod_{b\in B(a)} D(a, c, b) \longrightarrow B(a), \qquad (b, d) \longmapsto b. \tag{2}$$

A commuting square $J_{a,c} \to u$ consists of a map $k : B(a) \to X$ and, for each (b, d), an element of $Y_{k(b)}$. Therefore

$$\operatorname{dom} \widehat{P}(u) \cong \coprod_{a,c} \mathcal{A}(J_{a,c}, u), \qquad \operatorname{cod} \widehat{P}(u) \cong \coprod_{a} \mathbf{Set}(B(a), X). \tag{3}$$

The source-to-target arrow in (1) sends each square $J_{a,c} \to u$ to its bottom map $B(a) \to X$.

Lemma 2.2. *The endofunctor $\widehat{P}$ is an (internal) polynomial endofunctor of the locally cartesian closed category $\mathcal{A}$, with shapes the arrow $\sum_a C(a) \to A$ and with operation arities $J_{a,c}$. In particular, it is accessible and its free monad $\widehat{T} = \widehat{P}^*$ exists.*

Proof. Let $y(0) = (1 \xrightarrow{\mathrm{id}} 1)$ and $y(1) = (\varnothing \to 1)$ be the two covariant representables in $\mathcal{A} = \mathbf{Set}^{(0\to 1)}$. Define the arrow of shapes and the arrow of positions by

$$\mathsf{Sh} = \left(\coprod_a C(a) \longrightarrow A\right), \qquad \mathsf{Pos} = \left(\coprod_{a,c,b} D(a, c, b) \longrightarrow \coprod_a B(a)\right).$$

There is a morphism $\mathsf{Pos} \to \mathsf{Sh}$ in $\mathcal{A}$, whose upper map sends (a, c, b, d) to (a, c) and whose lower map sends (a, b) to a. Together with the unique maps to the terminal arrow, this is an internal polynomial diagram

$$1_{\mathcal{A}} \longleftarrow \mathsf{Pos} \longrightarrow \mathsf{Sh} \longrightarrow 1_{\mathcal{A}}.$$

A generalised shape $y(0) \to \mathsf{Sh}$ picks (a, c); pulling $\mathsf{Pos} \to \mathsf{Sh}$ back along it gives exactly $J_{a,c} : \coprod_b D(a, c, b) \to B(a)$. A shape $y(1) \to \mathsf{Sh}$ picks a; its pullback arity is $(\varnothing \to B(a))$. The dependent-product/pushforward description of this internal polynomial therefore has upper values $\coprod_{a,c} \mathcal{A}(J_{a,c}, u)$ and lower values $\coprod_a \mathbf{Set}(B(a), X)$, which are precisely (1) and (3). Every arity is a set, so a sufficiently large regular cardinal makes $\widehat{P}$ accessible. Its free monad exists by the free-algebra construction in the locally presentable category $\mathcal{A}$. □

2.1 The free certified programs

For $u : Y \to X$ and an ordinary tree $t \in T(X) = P^*(X)$, define the fibre $\mathrm{FreeDep}_{p,r}(u, t)$ recursively:

$$\mathrm{FreeDep}(u, \mathsf{return}(x)) = Y_x, \tag{4}$$

$$\mathrm{FreeDep}(u, \mathsf{call}(a, k)) = \coprod_{c\in C(a)}\ \prod_{b\in B(a)}\ \prod_{d\in D(a,c,b)} \mathrm{FreeDep}(u, k(b)). \tag{5}$$

Here $k : B(a) \to T(X)$ is a continuation family. This is the set-level reading of the constructors `returnD` and `bindD` in [1, §5.2].

Proposition 2.3 (Free-program comparison)**.** *There is a natural identification of arrows*

$$\widehat{T}(u) \cong \left(\coprod_{t \in T(X)} \mathrm{FreeDep}_{p,r}(u,t) \longrightarrow T(X) \right), \qquad T = P^*. \tag{6}$$

Under this identification, the unit and multiplication of $\widehat{T}$ are verified return and substitution: they map a proof of an input to a proof of its returned program, and substitute certified programs while threading all required witnesses. In particular $\mathrm{cod}\ \widehat{T} = T\ \mathrm{cod}$.

Proof. The initial-algebra recursion for free $\widehat{P}$-terms has two cases: variable leaves (giving (4)) and operation nodes (giving (5)). Forgetting proof data at every node gives exactly a free P-term, and the fibre over that term is specified by the displayed recursion. Free-monad unit and multiplication are respectively insertion of a leaf and substitution of trees, and their upper components are the induced proof-recursive operations. Well-founded induction on the trees (allowing ordinal ranks for infinitely branching operations) establishes naturality and the monad laws. There is also a categorical reason why no ordinary program is lost: the functor $\mathrm{cod} : \mathcal{A} \to \mathbf{Set}$ has a left adjoint $X \mapsto (\varnothing \to X)$ and a right adjoint $X \mapsto (X \xrightarrow{\mathrm{id}} X)$. Hence it preserves all limits and colimits, including the initial-algebra and transfinite-colimit constructions of free monads. Together with $\mathrm{cod}\ \widehat{P} = P\ \mathrm{cod}$, this yields $\mathrm{cod}\ \widehat{T} = P^*\ \mathrm{cod}$. □

3 Arrow state passing and a typed Yoneda calculation

Take a residual-state arrow $e : E \to S$. The relevant monad on $\mathcal{A}$ is

$$\mathrm{St}_e(\mathrm{Id})(u) = (e \times u)^e.$$

Unlike the naive pointwise exponential of arrows, this exponential has *different* upper and lower components. This is the main calculation needed for a correct certification theorem.

Lemma 3.1 (Exponential in the arrow category)**.** *For $e : E \to S$ and $u : Y \to X$, the exponential u^e is the arrow*

$$\mathcal{A}(e,u) \longrightarrow \mathbf{Set}(S,X)$$

sending a commuting square to its lower map. Consequently

$$\mathrm{St}_e(\mathrm{Id})(u) = \left(\mathcal{A}(e, e \times u) \longrightarrow \mathbf{Set}(S, S \times X)\right). \tag{7}$$

Proof. For any arrow $h : Z \to V$, a square $h \times e \to u$ consists of a lower map $V \times S \to X$ and an upper map $Z \times E \to Y$, with their compatibility equation. Currying gives respectively $V \to \mathbf{Set}(S,X)$ and $Z \to \mathcal{A}(e,u)$, compatible over h. This is precisely the exponential universal property in $\mathcal{A}$. Set $u \leftarrow e \times u$ for (7). □

Define the ordinary Mealy transition functor

$$F_p(S) = \prod_{a \in A} (B(a) \times S). \tag{8}$$

For a state arrow $e : E \to S$, put $E_s = \{z \in E \mid e(z) = s\}$ and define $\widehat{F}_r(e)$ to be the arrow

$$\underbrace{\coprod_{\delta \in F_p(S)} \prod_{a \in A} \prod_{c \in C(a)} \Big(D(a,c,b_a(\delta)) \times E_{s_a(\delta)} \Big)}_{\mathrm{dom}\ \widehat{F}_r(e)} \longrightarrow \underbrace{F_p(S)}_{\mathrm{cod}\ \widehat{F}_r(e)}. \tag{9}$$

Here $\delta_a = (b_a(\delta), s_a(\delta))$. The arrow forgets certified-transition data and retains δ; it is functorial in maps of state arrows.

Theorem 3.2 (Certified runner–coalgebra comparison)**.** *For every state arrow $e : E \to S$ there is a natural bijection*

$$\mathrm{Mnd}(\widehat{T}, \mathrm{St}_e(\mathrm{Id})) \;\cong\; \mathcal{A}(e, \widehat{F}_r(e)). \tag{10}$$

More explicitly, a certified runner is exactly:

(i) an ordinary Mealy transition $\delta_a : S \to B(a) \times S$ for every $a \in A$;

(ii) for every $z \in E$, $a \in A$ and $c \in C(a)$, a witness

$$d_{a,c}(z) \in D(a, c, b_a(e(z)))$$

and a certified successor $z'_{a,c} \in E$ such that

$$e(z'_{a,c}) = s_a(e(z)). \tag{11}$$

The response $b_a(e(z))$ and the underlying successor $s_a(e(z))$ are independent of the precondition witness c; only the guarantee witness and certified successor may vary with c. Runner morphisms correspond precisely to $\widehat{F}_r$-coalgebra morphisms. Hence there is an isomorphism of categories over $\mathcal{A}$,

$$\mathrm{Run}(\widehat{T}, \mathrm{Id}) \cong \mathrm{Coalg}(\widehat{F}_r). \tag{12}$$

Proof. The universal property of the free monad gives

$$\mathrm{Mnd}(\widehat{T}, \mathrm{St}_e(\mathrm{Id})) \cong \mathrm{Nat}(\widehat{P}, \mathrm{St}_e(\mathrm{Id})).$$

By (3), the lower component of a transformation $\widehat{P} \to \mathrm{St}_e(\mathrm{Id})$ is a family, indexed by a, of natural transformations

$$\mathbf{Set}(B(a), -) \longrightarrow \mathbf{Set}(S, S \times -).$$

By Yoneda these are functions $S \to S \times B(a)$, equivalently the ordinary transitions δ_a (with the factors permuted).

For every (a, c) the upper component restricts to a natural transformation

$$\mathcal{A}(J_{a,c}, -) \longrightarrow \mathcal{A}(e, e \times -).$$

By Yoneda again, this is a square $e \to e \times J_{a,c}$. Write its lower map as $s \mapsto (s_a(s), b_a(s))$, and its upper map as

$$z \longmapsto (z'_{a,c}, (b_{a,c}(z), d_{a,c}(z))).$$

To see the compatibility directly, the inclusion $y(1) \to y(0)$ induces an arity map $\iota_{a,c} : (\varnothing \to B(a)) \to J_{a,c}$. Yoneda gives, for each a, a lower element $n_a \in \operatorname{cod} \mathrm{St}_e(\mathrm{Id})(\varnothing \to B(a))$ and, for each (a, c), an upper element $m_{a,c} \in \operatorname{dom} \mathrm{St}_e(\mathrm{Id})(J_{a,c})$. Since a natural transformation of arrow-valued functors must commute with its structure arrows, $m_{a,c}$ projects to the image of n_a under $\mathrm{St}_e(\mathrm{Id})(\iota_{a,c})$. By Lemma 3.1, the lower component of $\mathrm{St}_e(\mathrm{Id})$ depends only on the codomain of its argument. The map $\mathrm{St}_e(\mathrm{Id})(\iota_{a,c})$ therefore acts as the identity on that component: every $m_{a,c}$ has the *same* lower datum n_a. Unwinding the upper square gives exactly $b_{a,c}(z) = b_a(ez)$ and $e(z'_{a,c}) = s_a(ez)$. Conversely, these data define compatible natural transformations on every representable summand, hence a transformation of arrow functors. This yields (10), whose right side is exactly the square $e \to \widehat{F}_r(e)$ by (9). Natural transformations between state arrows preserve these same equations, giving (12). □

Remark 3.3 (An explanation of the independence of preconditions). In Aberlé's coinductive presentation the response and the successor machine are fixed by the underlying Mealy object, before a precondition witness c is considered. Here this is *forced by naturality*: distinct witnesses c cannot change the lower runner because their upper arities map to the same lower arity ($\varnothing \to B(a)$). Thus the arrow-polynomial calculation explains the design of appD rather than merely matching its signature.

Remark 3.4 (What is and is not certified). The underlying machine is defined on *all* $s \in S$, while the upper transitions need be defined only for $z \in E$. The arrow $E \to S$ need not be surjective or monic: uncertified states and multiple certificates for one state are both allowed. A precondition $C(a) = \varnothing$ imposes no upper obligation for a, but it does not remove the underlying ordinary transition.

4 Measuring comonad and the final dependent Mealy object

Proposition 4.1 (Concrete measuring construction). *For small A, B, C, D, the functor $\widehat{F}_r : \mathcal{A} \to \mathcal{A}$ is accessible and admits a cofree comonad G_r. Consequently*

$$\mathrm{Coalg}(G_r) \simeq \mathrm{Coalg}(\widehat{F}_r) \simeq \mathrm{Run}(\widehat{T}, \mathrm{Id})$$

over $\mathcal{A}$. Thus G_r represents the runner category. If an ambient Sweedler hom $\{\widehat{T}, \mathrm{Id}\}$ is available with this runner classification, then there is a canonical isomorphism $G_r \cong \{\widehat{T}, \mathrm{Id}\}$.

Proof. Choose a regular cardinal larger than all the arities and witness-indexing sets in (9). The construction uses set-indexed sums, products, and pullbacks of uniformly bounded arity, so it preserves sufficiently filtered colimits. Hence $\widehat{F}_r$ is accessible on the locally presentable category $\mathcal{A}$. The category of coalgebras of an accessible endofunctor is locally presentable and its forgetful functor creates colimits; by the adjoint functor theorem it has a right adjoint. This is the cofree coalgebra functor, inducing G_r. The other equivalence is Theorem 3.2. The final assertion follows from uniqueness of a comonad representing the same category of coalgebras over $\mathcal{A}$. □

Let $M = \mathrm{Mealy}(p)$ be the final coalgebra for F_p, and write the final transition at $m \in M$ as

$$\mathrm{app}(m, a) = (b_a(m), m_a).$$

Define $\mathrm{Cert}(m)$ coinductively by

$$\mathrm{Cert}(m) \cong \prod_{a \in A} \prod_{c \in C(a)} (D(a, c, b_a(m)) \times \mathrm{Cert}(m_a)). \tag{13}$$

This is precisely the equation for Aberlé's $\mathrm{DepMealy}(p, r, m)$ [1, Def. 6.1] interpreted in sets.

Theorem 4.2 (Final certified Mealy machine). *The arrow*

$$\boxed{\coprod_{m \in M} \mathrm{Cert}(m) \longrightarrow M, \qquad (m, \gamma) \longmapsto m} \tag{14}$$

with its induced transitions is the final $\widehat{F}_r$-coalgebra. In particular it is the cofree measuring coalgebra on the terminal arrow $1 \to 1$. Its lower component is the final ordinary Mealy machine, and its fibre over m is the type of coinductive verification evidence for that behaviour.

Proof. A coalgebra $e : E \to S$ has an underlying F_p-coalgebra on S, hence a unique behaviour map $\mathrm{Beh} : S \to M$. For each $z \in E$, coiterate the certified transitions from Theorem 3.2: at (a, c) record $d_{a,c}(z)$ and continue with $z'_{a,c}$. The compatibility (11) ensures the resulting evidence belongs to $\mathrm{Cert}(\mathrm{Beh}(ez))$ rather than to a fibre over an unrelated behaviour. This gives an arrow morphism

$$\begin{array}{ccc} E & \longrightarrow & \coprod_m \mathrm{Cert}(m) \\ {\scriptstyle e}\downarrow & & \downarrow \\ S & \xrightarrow[\mathrm{Beh}]{} & M \end{array}$$

which preserves coalgebra structure by construction. Any other coalgebra morphism has the same lower component by finality of M; its upper component is forced by the coinductive equation (13), proving uniqueness. A cofree coalgebra on the terminal object is final, giving the last identification. □

5 Verified implementations as tight interpretations

Let (p, r) be a high-level specification and (q, s) a low-level specification, with certified polynomials $\widehat{P}_p, \widehat{P}_q$. An ordinary implementation is a family

$$f_a \in T_q(B_p(a)) \qquad (a \in A_p), \qquad T_q = P_q^*. \tag{15}$$

Equivalently, $f : P_p \to T_q$ is a polynomial natural transformation, extending uniquely to a monad morphism $f^* : T_p \to T_q$. The superscript here denotes *free extension*, not the pullback of worlds.

Definition 5.1 (Verified implementation)**.** A verification of f comprises, for each $a \in A_p$ and $c \in C_p(a)$, an element

$$\widehat{f}_{a,c} \in \mathrm{FreeDep}_{q,s}(J^p_{a,c}, f_a), \tag{16}$$

where $J^p_{a,c} : \sum_{b \in B_p(a)} D_p(a, c, b) \to B_p(a)$ is the proof-bearing return type from (2). This is the set-level version of Aberlé's dependent morphism $\Rightarrow$ Dep for verified program implementations [1, Def. 5.2].

Proposition 5.2 (Free extension of verification)**.** *Giving an ordinary implementation f and verified witnesses (16) is equivalent to giving a natural transformation*

$$\widehat{f} : \widehat{P}_p \longrightarrow \widehat{T}_q$$

of arrow endofunctors, whose codomain component is $f : P_p \to T_q$. It extends uniquely to a monad morphism

$$\widehat{f}^\sharp : \widehat{T}_p \longrightarrow \widehat{T}_q \tag{17}$$

lying over $f^* : T_p \to T_q$. Composition of verified implementations is carried to composition of these monad morphisms.

Proof. Use the representable decomposition (3) for $\widehat{P}_p$. The lower Yoneda datum at a is $f_a \in T_q(B_p(a))$, while the upper Yoneda datum at (a, c) is an element of $\widehat{T}_q(J^p_{a,c})$ projecting to f_a. Proposition 2.3 identifies its fibre with (16). Naturality forces all witnesses indexed by c to project to the *same* f_a. Free-monad extension gives (17), and uniqueness of the extension and monad associativity identify verified substitution with composition. □

Theorem 5.3 (Certified induction is comodel restriction)**.** *A verified implementation $\widehat{f}$ induces a functor*

$$\widehat{f}^* : \mathrm{Coalg}(\widehat{F}_s) \longrightarrow \mathrm{Coalg}(\widehat{F}_r) \tag{18}$$

which preserves the residual state arrow $e : E \to S$. It is defined by precomposing runners with $\widehat{f}^\sharp$. It commutes with the forgetful functors to ordinary Mealy coalgebras, where the induced functor is implementation execution along f. On final behaviours the resulting map agrees, by the return/call recursion, with Aberlé's `prog→mealyD` *construction [1, Prop. 6.2].*

Proof. For a certified q-machine, Theorem 3.2 supplies a monad map $\beta : \widehat{T}_q \to \mathrm{St}_e(\mathrm{Id})$. Precomposition gives $\beta\widehat{f}^\sharp : \widehat{T}_p \to \mathrm{St}_e(\mathrm{Id})$, and Theorem 3.2 turns this into a certified p-machine on the same e. The codomain component is the analogous composite $T_p \to T_q \to \mathrm{St}_S(\mathrm{Id})$, so ordinary semantics commutes with forgetting. On a returned value, the induced certified evaluator just returns that value and its witness; on a call, it executes the target certified transition, obtains a guarantee witness, and recurses on the corresponding certified continuation. These are exactly the two recursive clauses defining the evaluation of FreeDep, and hence the semantic operation called `run-mealyD`. Coiteration over successive source commands yields `prog→mealyD`. Monad associativity proves compositionality independently of any particular program syntax. □

Corollary 5.4 (Binary sums of independent certified machines)**.** *Let $e_p : E_p \to S_p$ and $e_q : E_q \to S_q$ be certified machines for (p, r) and (q, s). Their product arrow $e_p \times e_q$ carries a canonical certified machine structure for the coproduct interface $p \oplus q$ with its coproduct specification. A command from the left summand acts on the p-component and leaves the q-component fixed, and conversely on the right. Explicitly, the upper transition at (z_p, z_q) for a left command (a, c) returns the original guarantee witness and $(z'_{p,a,c}, z_q)$; its lower component returns the response and $(s'_{p,a}, s_q)$. These equations make the certification square commute. On final behaviours this is Aberlé's* `depMealy⊕` *(Prop. 6.3). This concerns independent components; it does not imply a general tensor product of arbitrary certified runners.*

6 Invariants, genuinely proof-relevant certificates, and equations

6.1 The invariant state interface

Consider the state signature

$$\mathsf{get} : 1 \to \mathbb{N}, \qquad \mathsf{put}(n) : 1 \to 1 \quad (n \in \mathbb{N}).$$

Let $Q(n)$ be a set of admissible certificates for n. The certified-state arrow is

$$e : E = \coprod_{n \in \mathbb{N}} Q(n) \longrightarrow \mathbb{N}, \qquad (n, q) \longmapsto n.$$

The dependent specification is

$$C(\mathsf{get}) = 1, \qquad D(\mathsf{get}, *, n) = Q(n),$$
$$C(\mathsf{put}(n)) = Q(n), \qquad D(\mathsf{put}(n), q, *) = 1.$$

On (n, q), get returns the response n and its evidence q, leaving the certified state unchanged. A $\mathsf{put}(n')$ call supplied with $q' \in Q(n')$ moves the certified state to (n', q'). This is Aberlé's `Invariant` construction in [1, §6.2]. For the special choice $Q(n) = \{n \text{ is even}\}$, these witnesses are propositions and are at most singleton-valued.

As a small *additional* example, introduce a high-level command $\mathsf{tick} : 1 \to \mathbb{N}$ with precondition $C(\mathsf{tick}) = 1$ and guarantee $D(\mathsf{tick}, *, k) = Q(k)$, using the evenness predicate. Implement it by

$$\mathsf{get}; \quad \mathsf{put}(n+2); \quad \mathsf{return}(n+2).$$

The ordinary response is $n+2$ and the successor is $n+2$. Here the verified implementation fibre is calculated explicitly:

$$\mathrm{FreeDep}(J_{\mathsf{tick},*}, f_{\mathsf{tick}}) \cong \prod_{n \in \mathbb{N}} \prod_{q \in Q(n)} \left(\coprod_{t \in Q(n+2)} Q(n+2) \right). \tag{19}$$

Indeed, get supplies a response n and evidence q; calling $\mathsf{put}(n+2)$ requires an assumption $t \in Q(n+2)$, and the final $\mathsf{return}(n+2)$ requires a possibly distinct return certificate in $Q(n+2)$. The canonical choice sends q to (q^+, q^+), where q^+ is the witness that $n+2$ is even. The two appearances of $Q(n+2)$ distinguish the assumption consumed by `put` from the guarantee returned to the caller. This illustrates FreeDep, but being proposition-valued it does *not* yet exhibit multiple certificates for the same ordinary state.

6.2 A genuinely proof-relevant example: sums of two squares

Keep the state interface, but now define

$$\boxed{Q(n) = \{(x, y) \in \mathbb{Z}^2 \mid x^2 + y^2 = n\}.}$$

This is not a truth-valued predicate: the same number can have multiple distinct certificates; for example $(0, 5)$ and $(3, 4)$ both belong to $Q(25)$. The same $e : E \to \mathbb{N}$ and the same certified $\mathsf{get}/\mathsf{put}$ operations therefore retain genuinely different witnesses over one state.

For each $m \in \mathbb{N}$ introduce a high-level operation $\mathsf{mul}(m)$ with response set $\mathbb{N}$, precondition and guarantee

$$C(\mathsf{mul}(m)) = Q(m), \qquad D(\mathsf{mul}(m), c, k) = Q(k).$$

The low-level program is

$$\mathsf{get}; \qquad \mathsf{put}(nm); \qquad \mathsf{return}(nm), \tag{20}$$

where n is the value returned by get. For each fixed input witness $c \in Q(m)$, its dependent free-program fibre is

$$\mathrm{FreeDep}(J_{\mathsf{mul}(m),c}, f_{\mathsf{mul}(m)}) \cong \prod_{n \in \mathbb{N}} \prod_{q \in Q(n)} \left(\coprod_{t \in Q(nm)} Q(nm) \right). \tag{21}$$

Write the get-witness $q = (x, y)$ and the caller's precondition witness $c = (u, v)$. The Brahmagupta–Fibonacci identity

$$(x^2 + y^2)(u^2 + v^2) = (xu - yv)^2 + (xv + yu)^2$$

produces an explicit certificate

$$q \odot c = (xu - yv,\ xv + yu) \in Q(nm). \tag{22}$$

Choose $t = q \odot c$ as the precondition for $\mathsf{put}(nm)$ and again as the certificate of the returned result. Thus the witness of (21) is

$$(n, q) \longmapsto (q \odot c,\ q \odot c).$$

The induced certified high-level machine carries

$$(n, q),\ m,\ c \in Q(m) \quad \longmapsto \quad (nm,\ q \odot c,\ (nm, q \odot c)).$$

Here the supplied precondition c is actually *used*. Different choices of c, or different representations of the same n, can yield different postcondition witnesses while the ordinary response and successor remain nm. This is the extra proof-relevant information retained in the fibres of the final certified-comodel object.

6.3 What changes when the state laws are imposed?

The preceding theorem is for the *free* state signature, without equations. The ordinary state theory includes laws such as "put s and then get" behaving like "put s and return s." This lower-level equality does not, by itself, identify proof witnesses upstairs.

A *certificate-stable lift* of this law can require that, after $\mathsf{put}(s)$ supplied with $t \in Q(s)$, a subsequent certified get return *that same witness* t, rather than merely some $t' \in Q(s)$. For the standard certified state machine above the law holds by construction. When $Q(s)$ has multiple elements, however, it is a genuine constraint on possible upper transitions: the underlying state equation alone permits an implementation that changes the certificate while preserving the same ordinary state.

To formalise this at the theory level one should choose a congruence on $\widehat{T}$ lying over the ordinary state-theory congruence and explicitly specify which proof-bearing programs are equated. Comodels of that presented certified theory would be the certified free comodels satisfying those equations. *We do not claim* that every ordinary equation has a canonical or automatic proof-relevant lift, or that this general equational measuring theorem has been proved here.

7 World-typed machines and precise scope

There are two independent uses of "dependent" in this programme. Above, the *verification witnesses* vary over commands and returned responses, while all operational states lie in one ordinary set. Crucially, $C(a)$ cannot inspect the machine state: it depends on the input command a, not on $s \in S$. A state-dependent partial operation such as `pop` only on nonempty stacks therefore cannot be specified by choosing $C(\mathsf{pop})$ alone. Restricting certified states to nonempty stacks does not resolve the issue: popping a singleton stack reaches the empty stack, so such a choice is not transition-closed. In the world-typed gallery, commands and continuation states themselves vary over worlds W, providing the necessary distinction.

For example, let $W = \mathbb{N}$ record stack length. In world n, `push` operations lead to world $n + 1$, while `pop` is offered only when $n > 0$ and leads to $n - 1$. One can take the concrete

states $S_n = V^n$ for an alphabet V, and optionally attach certified states $E_n \to S_n$ in each world. The type of legal operations now depends on the abstract state, not merely on a fixed global interface. This makes the world-typed extension a *necessary next step* for such protocols, rather than an inessential embellishment.

For a world-typed interface take A_w, $B_w(a)$ and destinations $d(w, a, b) \in W$. Its ordinary deterministic Mealy functor on $\mathbf{Set}^W$ is

$$F_W(S)_w = \prod_{a \in A_w} \coprod_{b \in B_w(a)} S_{d(w,a,b)}.$$

A certified state is now an arrow $e = (E_w \to S_w)_{w \in W}$ in $(\mathbf{Set}^W)^{\to}$. Adding $C(w, a)$ and $D(w, a, c, b)$ yields an explicit certified transition functor whose component at w is the arrow

$$\coprod_{\delta \in F_W(S)_w} \prod_{a \in A_w} \prod_{c \in C(w,a)} \Big(D(w, a, c, b_a(\delta)) \times E_{d(w,a,b_a(\delta)),\, s_a(\delta)} \Big) \longrightarrow F_W(S)_w. \tag{23}$$

Here $E_{v,s}$ denotes the fibre of e_v over $s \in S_v$. Its coalgebras are *certified world-typed Mealy machines*, with evidence-bearing transition squares that respect response-dependent destination worlds. The same representable/Yoneda calculation applies using the indiscrete world-change target monad K_W and its arrowwise lifting, rather than Id (which cannot implement a forced change of world). Formula (23) is therefore a concrete example of combining both forms of dependence, but it does *not* establish a universal measuring theorem for arbitrary span runners between different implementation world types.

Certification and a viability fixed point

For a fixed ordinary Mealy coalgebra $\delta : S \to F_p(S)$, define the monotone operator

$$\Psi_\delta(U) = \{ s \in S \mid \forall a \in A \; \forall c \in C(a), \quad D(a, c, b_a(s)) \neq \varnothing \text{ and } s_a(s) \in U \}.$$

Assuming ordinary choice, the states supporting some coinductive certificate are precisely the greatest fixed point $\nu\Psi_\delta$. For the "only if" direction, the support of the coinductive certification family is post-fixed; for the converse, choose a guarantee witness for each (s, a, c) in a post-fixed set and coiterate along $s \mapsto s_a(s)$. This is the certification analogue of the viability-kernel calculation in the elementary dependent-runner gallery. It also illustrates a useful limit of the present fragment: viability depends on the underlying transitions, while the source-side operation assumptions themselves cannot inspect the current state without an additional world index.

The higher-type-theoretic version also requires further hypotheses: $\mathbf{Type}^{\to}$ can retain homotopical identifications of proofs that $\mathbf{Set}^{\to}$ does not. Nor have we established that every accessible monad on $\mathbf{Set}^{\to}$ lies in a closed duoidal universe with all the abstract Sweedler measuring operations. Our theorem is stronger in one direction and narrower in another: for the displayed *free certified polynomial* theories, the representing coalgebra comonad and the verified-program action are constructed explicitly.

8 The category-valued comparison and related work

Aberlé's categorical perspective in [1, §7] organises ordinary and dependent Mealy behaviours into set-valued contravariant presheaves on interface and specification categories,

related by a monoidal natural transformation. Our result retains *whole categories* of state-bearing coalgebras instead of only the sets of final behaviours. More precisely, associate to a specification (p, r) the category $\mathrm{Coalg}(\widehat{F}_r)$ and to an interface p the category $\mathrm{Coalg}(F_p)$. The codomain functor induces a forgetful functor

$$\mathrm{Coalg}(\widehat{F}_r) \longrightarrow \mathrm{Coalg}(F_p), \qquad (E \xrightarrow{e} S, \widehat{\delta}) \longmapsto (S, \delta).$$

Theorem 5.3 shows that verified implementations act contravariantly on these categories, compatibly with ordinary implementations and their composition. With the usual identifications of syntactic substitution, this is a category-valued (or pseudofunctorial) refinement of the set-valued operational semantics. The final object of $\mathrm{Coalg}(F_p)$ is $\mathrm{Mealy}(p)$; the final object of $\mathrm{Coalg}(\widehat{F}_r)$ lies above it and has fibre $\mathrm{DepMealy}(p, r, m)$, by Theorem 4.2. A general restriction functor need not preserve final objects, but its value at a final q-coalgebra has a unique map to the final p-coalgebra, supplying the usual induced map on final behaviours. Arbitrary arrows $E \to S$, including those with uncertified states and multiple certifications, constitute the extra categorical information.

The relationship to three other traditions helps locate the result. Ahman–Bauer's runners [4] motivate the operational reading of a Mealy machine as a resource interpreter; the McDermott–Rivas–Uustalu monad-map formulation [2] is the precise notion of runner used here. Libkind–Spivak [5] describe program execution through the module interaction of a free-monad construction and a cofree-comonad construction; our explicit cofree comonad G_r and certified restriction functors suggest a proof-relevant analogue of that interaction, but we have not identified an entire dependent module structure. Finally, the codomain fibration $\mathrm{cod} : \mathbf{Arr}(\mathbf{Set}) \to \mathbf{Set}$ and the refinement of inductive/coinductive types place the comparison beside fibrational methods of Hermida–Jacobs [6], Atkey–Johann–Ghani [7] and Ghani–Johann–Fumex [8]. Our $\widehat{F}_r$ is a *specification-indexed* lifting, not asserted to be a canonical predicate lifting.

The ambient picture also has a simple explanation. Operational worlds indexed by a discrete W use $\mathbf{Set}^W$, certificates use $\mathbf{Set}^{(0\to1)}$, and their combination is

$$(\mathbf{Set}^W)^{(0\to1)} \cong \mathbf{Set}^{W\times(0\to1)}.$$

Every evaluation functor in a presheaf category has both Kan-extension adjoints. This suggests testing polynomial Sweedler runners in presheaf toposes with genuinely non-discrete world categories, before attempting a completely ambient-independent dependent double category. Existence and compatibility of the necessary measuring structures remain hypotheses to verify.

9 Conclusion

The comparison gives a small, checkable entry point to the hoped-for ambient-independent theory. At set level:

$$(p, r) \longmapsto \widehat{P} \longmapsto \widehat{T} = \widehat{P}^*,$$
$$\mathrm{Mnd}(\widehat{T}, \mathrm{St}_e(\mathrm{Id})) \cong \mathcal{A}(e, \widehat{F}_r(e)),$$
$$\mathrm{Coalg}(G_r) \cong \mathrm{Coalg}(\widehat{F}_r),$$
$$G_r(1 \to 1) = \left(\coprod_m \mathrm{DepMealy}(p, r, m) \to \mathrm{Mealy}(p) \right).$$

Verified program morphisms extend to monad morphisms of certified theories; restriction of comodels realises their certified operational semantics. The statements concern *genuine proof witnesses*, as the sums-of-two-squares example demonstrates; ordinary Boolean invariants are only a special case. The remaining ambitious task is to prove the compatibility of these constructions with changes of ambient category and world object, and ultimately with span-valued runners, without losing the elementary verified-Mealy calculation that motivates them.

References

[1] C. B. Aberlé, *Compositional Program Verification with Polynomial Functors in Dependent Type Theory*, arXiv:2604.01303 (2026), esp. §§5–6. https://arxiv.org/abs/2604.01303.

[2] D. McDermott, E. Rivas and T. Uustalu, *Sweedler Theory of Monads*, FoSSaCS 2022, LNCS 13242, pp. 428–448. https://doi.org/10.1007/978-3-030-99253-8_22.

[3] J. Adámek and J. Rosický, *Locally Presentable and Accessible Categories*, Cambridge University Press, 1994.

[4] D. Ahman and A. Bauer, *Runners in Action*, ESOP 2020, LNCS 12075, pp. 29–55. https://doi.org/10.1007/978-3-030-44914-8_2.

[5] S. Libkind and D. I. Spivak, *Pattern Runs on Matter: The Free Monad Monad as a Module over the Cofree Comonad Comonad*, ACT 2024, arXiv:2404.16321. https://arxiv.org/abs/2404.16321.

[6] C. Hermida and B. Jacobs, *Structural Induction and Coinduction in a Fibrational Setting*, Information and Computation 145 (1998), no. 2, pp. 107–152. https://doi.org/10.1006/inco.1998.2725.

[7] R. Atkey, P. Johann and N. Ghani, *Refining Inductive Types*, Logical Methods in Computer Science 8(2:9), 2012. https://doi.org/10.2168/LMCS-8(2:9)2012.

[8] N. Ghani, P. Johann and C. Fumex, *Indexed Induction and Coinduction, Fibrationally*, Logical Methods in Computer Science 9(3:6), 2013. https://doi.org/10.2168/LMCS-9(3:6)2013.