The Two-Dimensional Algebra of Runners

Stage A.1–A.7: motivation, mathematical synthesis and proof audit

14 October 2026

Abstract

A computational theory specifies which operations may be performed and which equations they obey; a concrete machine must realise those operations using its own state and possibly a different repertoire of effects. The problem of Stage A is to understand these *stateful implementations*: how to classify and compose them, how their behaviour induces semantics for programs and handlers, and what it means to compare two implementations. We use Set-based runners to give explicit answers. The route runs from behaviour categories and, in polynomial cases, presheaf topoi, through horizontal execution and its residuals, to induction–delegation and effectful 2-cells. A freeness theorem determines when monoid coinduction is itself runner delegation; left-free bisets explain coherence for writer runners; and a finite-powerset example tests exact and lax comparison beyond polynomial residues. The core calculations are proved, and the remaining boundaries are stated. The result is a stable Set-based starting point for the full Sweedler adjunctions, not an assertion that every ambient category has the same properties.

Contents

Introduction: the problem behind Stage A **2**

Scope, conventions, and status **3**

A.1: Three writer-to-unary runner topoi **4**

A.2: Cartesian structure, generators, and logic **5**

A.3: Horizontal composition and two residuals **6**

A.4: Handlers, induction, and a phase adjunction **8**

A.5: The freeness criterion for monoid coinduction **9**

A.6: Left-free bisets and complete handler semantics **10**

A.7: Nondeterministic and lax comparisons **14**

Stocktaking: what problem Stage A has solved **15**

Introduction: the problem behind Stage A

From a theory of operations to an implementation

A theory of computational effects describes the operations a program is entitled to request, including the equations that make different programs equivalent. That description does not, by itself, say *how* a particular machine executes the operations. A logical operation might require several physical steps; its implementation may depend on a hidden control state; and the machine may leave behind work to be performed in a different language of effects. To reason compositionally about systems, we need to retain all three ingredients: the operations requested, the machine state, and the residual computation.

A stateful runner from a theory T to a residual theory R is exactly such an implementation. On a set S of internal states it assigns to each T-computation an R-computation, returning both a value and an updated state:

$$\rho_X : TX \times S \longrightarrow R(X \times S).$$

The runner laws ensure that operations identified by the equations of T receive the same implementation; we therefore work directly with the law-governed, possibly quotient, theory rather than first collecting arbitrary state machines and then filtering them. We are not merely comparing two signatures, nor identifying a source command with a single target command: one implementation can issue a sequence of residual operations while evolving its own state.

The most elementary running example already exhibits the difficulty. Take $T = R = \mathbb{N}y$, the unary "tick" theory, with $\mathbb{N}y(X) = \mathbb{N} \times X$. A runner is a set S, an update $f : S \to S$, and a cost $k : S \to \mathbb{N}$. One requested tick advances the hidden state to $f(s)$ and performs $k(s)$ residual ticks. After n requests, the emitted cost is

$$K_s(n) = \sum_{i=0}^{n-1} k(f^i(s)), \qquad \rho_X((n, x), s) = \big(K_s(n), (x, f^n(s))\big).$$

For example, a two-phase machine that alternates states $0, 1$ with $k(0) = 0$, $k(1) = 2$ answers successive logical ticks with costs $0, 2, 0, 2, \ldots$. The source theory is unchanged; it is the implementation, not the language, that remembers the phase. Such examples may be read as schedulers, step refinements or resource-accounting machines. They also indicate why studying only stateless monad morphisms would miss much of the dynamics.

Four questions that the mathematics must answer

There is an underlying progression of problems, rather than a collection of unrelated category-theoretic constructions.

1. **Classification and observable behaviour.** When do two internal states have the same observable pattern of residual effects? What is the category of all implementations of fixed T in fixed R? Equations such as commutation should constrain complete state evolution, not merely cumulative output counts.
2. **Composition and universal constructions.** If one machine implements T using R, and a second implements R using Q, what is their composite T-to-Q machine? Is it different from making two independent observations of the same behaviour? What, if anything, is universal about reversing or resolving such an execution?
3. **Semantics for programs and handlers.** A machine should act not only on transitions but also on algebras interpreting the residual operations. Which handler transformations

result, and when can a right adjoint to an algebraic interpretation itself be implemented by a runner?

4. **Comparison of implementations.** Strict maps of hidden states are often too restrictive: changing a representation of the state can require a compensating residual effect. What are the correct 2-cells between runners, how do they compose, and what survives if residual effects are nondeterministic and comparison laws hold only by inclusion?

The safety and verification motivation is implicit in each question. Behavioural equivalence and simulation are meaningful only once we know which observations and costs are preserved; composing certified components requires a composition law for implementations; and handler semantics is how a theory of operations is interpreted against concrete observations. This note establishes the mathematical infrastructure for those tasks. It does not claim to derive a full verification calculus from the constructions below.

Why the seven parts appear in this order

A.1–**A.2** begin with runners between writer theories, where state updates, accumulated costs and behaviour categories can all be calculated. They show what the *local category of machines* contains and what its topos structure means. **A.3** then changes perspective: instead of studying machines at fixed source and target, we execute one machine through another. This creates a horizontal product, distinct from a Cartesian product, with its own adjoints and behavioural kernel. **A.4**–**A.5** pass from machines to their action on algebras and handlers, first exhibiting coinduction by phase runners and then explaining precisely when such an implementation is possible. **A.6** asks what maps between implementations must be added to obtain genuinely two-dimensional composition; the answer is effectful handler comparison, with left-free bisets as a concrete model. Finally, **A.7** tests the theory where the polynomial/topos picture no longer applies unchanged, and where inequalities rather than equalities express useful comparisons.

In short, we first ask *what machines are*, then *how they execute one another*, then *what they do to program semantics*, and finally *how they may be compared*. The calculations are arranged as answers to those questions.

Scope, conventions, and status

Write $T_M(X) = M \times X$ for the writer monad of a monoid M. A runner from T to R with state set S is a natural state-passing interpretation

$$\rho_X : TX \times S \longrightarrow R(X \times S)$$

compatible with the two monad structures. Composition $G \odot F$ means: execute F first, then use G to execute its residual computation. In a Set-based accessible context the strict local category is

$$\mathcal{H}(T, R) := \mathrm{Run}(T, R) \simeq \mathrm{Coalg}(\{T, R\}).$$

For writer monads we freely identify T_M-algebras with *left* M-sets and use *right* monoid actions on runner states. Opposite conventions reverse some displayed orders.

There are three levels of status: **calculated** (proofs below), **structural** (a theorem with stated hypotheses, for which a careful coherence presentation remains advisable), and **open** (a proposed extension). No claim of originality is made merely on account of its appearance in this note; the general Sweedler and comodel framework is due to the literature cited at the end [1, 2].

Part	Focus	Assessment
A.1	Writer-to-unary hom-topoi, behaviours and equations	Calculated
A.2	Representables, products, exponentials and internal truth values	Calculated
A.3	Horizontal residuals, phase machines and kernels	Calculated / structural
A.4	Induction, delegation, coinduction through phase runners	Calculated
A.5	Freeness criterion for coinduction along monoid maps	Theorem, proved
A.6	Left-free bisets and effectful handler comparisons	Theorem; biset coherence
A.7	Finite-powerset exact and lax tests	Calculated

A.1: Three writer-to-unary runner topoi

The first problem is to *classify* the possible implementations, not merely to describe an individual state machine. If the source operations form a monoid M and residual effects count ticks in $\mathbb{N}$, every implementation carries an accumulated-cost function satisfying an equation. The striking point is that the behaviour of each state can be used as an object of a small action category, and a whole runner then becomes a functor out of that category. This gives a concrete answer to the question of what the local category of implementations looks like, while keeping the equations of the source theory visible.

The general cost-cocycle construction

Let M be a monoid and take residual monad $T_{\mathbb{N}} = \mathbb{N} \times -$, with additive notation for $\mathbb{N}$. A runner $T_M \rightsquigarrow T_{\mathbb{N}}$ consists of a right M-action $s \cdot m$ on S and an $\mathbb{N}$-valued cocycle

$$c(s, 1) = 0, \qquad c(s, mn) = c(s, m) + c(s \cdot m, n). \tag{1}$$

The runner acts by $\rho_X((m, x), s) = (c(s, m), (x, s \cdot m))$. Strict maps $h : S \to S'$ preserve the action and the cocycle.

Set

$$B_M = \{K : M \to \mathbb{N} \mid K(1) = 0,\ K(mn) \geq K(m) \text{ for all } m, n\}.$$

For $m \in M$ let $(\sigma_m K)(n) = K(mn) - K(m)$. These shifts satisfy $\sigma_n(\sigma_m K) = \sigma_{mn} K$, in the convention that the action is on the right. Define $\mathcal{B}_M = B_M \rtimes M$: its objects are $K \in B_M$, with an arrow $(K, m) : K \to \sigma_m K$, composition given by multiplication in M.

Proposition A.1.1. *There is an equivalence*

$$\mathrm{Run}(T_M, T_{\mathbb{N}}) \simeq [\mathcal{B}_M, \mathbf{Set}],$$

and the associated measuring comonad on Set has underlying endofunctor

$$\{T_M, T_{\mathbb{N}}\}(X) \cong \coprod_{K \in B_M} X^M.$$

Its counit evaluates at $1 \in M$*; its comultiplication sends* (K, u) *to* $\big(K, m \mapsto (\sigma_m K, n \mapsto u(mn))\big)$.

Proof. For a runner define $K_s(m) = c(s, m)$. Equation (1) gives $K_{s \cdot m} = \sigma_m K_s$. Hence the fibre $F(K) = \{s \mid K_s = K\}$ carries action maps $F(K) \to F(\sigma_m K)$. Conversely, a covariant functor yields its disjoint union of fibres and the cocycle $c(s, m) = K(m)$ on the fibre over K. The constructions are inverse, including on strict runner maps. Encoding at a state the behaviour K and all successor states $s \cdot m$ gives the stated polynomial comonad and its structure maps. □

The category on the right is a Grothendieck topos (with the *covariant* copresheaf convention). This is an explicit instance of the polynomial-comonad/category correspondence, but the proof does not require a blanket assertion that arbitrary measuring comonads are polynomial.

The three specialisations

The following progression is deliberate. One generator produces a timeline of costs; two free generators produce a branching history tree; imposing commutation turns that tree into a grid and introduces a compatibility law. Thus passing from the free theory to a quotient changes not only the set of permitted observations but also the state transitions that a legitimate implementation must satisfy.

(i) $M = \mathbb{N}$: a runner is (S, f, k) with $f : S \to S$, $k : S \to \mathbb{N}$; its behaviour is the stream $a_s = (k(f^n(s)))_{n\geq 0}$. Thus $B_M \cong \mathbb{N}^{\mathbb{N}}$ and $\mathcal{B}_M$ is the stream-shift action category.

(ii) $M = A^*$ for $A = \{a, b\}$: a runner has arbitrary transitions f_a, f_b and costs k_a, k_b. Its behaviour is an edge labelling of the free binary history tree, an element of $\mathbb{N}^{A^+}$. The behaviour category is $\mathbb{N}^{A^+} \rtimes A^*$.

(iii) $M = \mathbb{N}^2$: a runner has commuting transitions f, g and costs k, ℓ satisfying

$$fg = gf, \qquad k(s) + \ell(f(s)) = \ell(s) + k(g(s)).$$

Its behaviour is a coordinatewise monotone potential $K : \mathbb{N}^2 \to \mathbb{N}$, $K(0, 0) = 0$. Edge labels are differences of the potential. For example $K(p, q) = p + q + pq$ produces costs $q + 1$ and $p + 1$: commutativity does not force independent costs.

The quotient $q : A^* \twoheadrightarrow \mathbb{N}^2$ restricts behaviours and identifies history arrows. Agreement of cumulative costs alone does *not* force equality of state transitions along ab and ba. The arrow quotient is therefore as important as the restriction of objects.

A.2: Cartesian structure, generators, and logic

Having represented runners as functors, we should ask what this category actually lets us *do*. Its representables are elementary behaviour probes: they present a prescribed cost stream and all its future shifts. Products then ask when two probes can describe *the same* evolving state; exponentials encode transformations compatible with arbitrary future shifts; and internal truth values describe properties stable under those shifts. These constructions are useful for behavioural reasoning, but they refer to a *fixed* source and residue and should not be confused with composing implementations.

From here until §A.5, set $\mathcal{H} = \mathrm{Run}(\mathbb{N}y, \mathbb{N}y) \simeq [\mathcal{B}, \mathbf{Set}]$ with $\mathcal{B} = \mathbb{N}^{\mathbb{N}} \rtimes \mathbb{N}$; write σ for stream shift.

Proposition A.2.1. *The representable* $\mathsf{y}_a = \mathcal{B}(a, -)$ *is the runner on* $\mathbb{N}$ *with transition* $n \mapsto n + 1$ *and cost* a_n*. Moreover* $\mathcal{H}(\mathsf{y}_a, S) \cong S(a)$*.*

Proof. Yoneda, together with the explicit description of the arrows issuing from a. □

For the two-phase machine in the introduction, take $a = (0, 2, 0, 2, \ldots)$. The representable y_a is a universal probe for that complete observable cost pattern: a map $\mathsf{y}_a \to S$ picks out a state of S with behaviour a, even when S has many additional hidden states or different internal organisation. This explains the value of representables before we invoke more elaborate topos constructions.

The Cartesian product of F, G has states (s, t) with *equal complete cost streams*; it is not their unrestricted state product. For the constant-zero stream $\underline{0}$, the product $\mathsf{y}_{\underline{0}} \times \mathsf{y}_{\underline{0}}$ has states $(n, m) \in \mathbb{N}^2$ and diagonal successor $(n+1, m+1)$. Its connected rays are indexed by $m-n \in \mathbb{Z}$, so

$$\mathsf{y}_{\underline{0}} \times \mathsf{y}_{\underline{0}} \cong \coprod_{d \in \mathbb{Z}} \mathsf{y}_{\underline{0}}. \tag{2}$$

For arbitrary a, b, the product of representables has states (n, m) with $\sigma^n a = \sigma^m b$; its rays are the minimal compatible synchronisations.

For the Cartesian exponential,

$$(G^F)(a) \cong \mathrm{Nat}(\mathsf{y}_a \times F, G). \tag{3}$$

At the constant-zero stream, (2) gives

$$(\mathsf{y}_{\underline{0}}^{\mathsf{y}_{\underline{0}}})(\underline{0}) \cong \prod_{d \in \mathbb{Z}} \mathcal{H}(\mathsf{y}_{\underline{0}}, \mathsf{y}_{\underline{0}}) \cong \mathbb{N}^{\mathbb{Z}}.$$

Equivalently an element is $h : \mathbb{N}^2 \to \mathbb{N}$ satisfying $h(n+1, m+1) = h(n, m) + 1$. This is far larger than the set of ordinary endomorphisms of $\mathsf{y}_{\underline{0}}$, which is $\mathbb{N}$.

The subobject classifier at a consists of cosieves (right-closed sets of outgoing arrows), namely

$$\Omega(a) = \{\varnothing, \uparrow 0, \uparrow 1, \ldots\}, \quad \uparrow r = \{n \in \mathbb{N} : n \geq r\}.$$

For a forward-invariant subset $P \subseteq S$, its characteristic arrow sends s to the cosieve $\{n : f^n(s) \in P\}$: it records the first entry time into P, or ∞. This is an interpretation of *internal truth values*, not an identification of the entire topos logic with an external temporal-logic syntax. In particular, the classifier packages a persistence property and a first-entry time, illustrating how logical structure can retain dynamical information without providing, on its own, a complete safety or temporal-logic calculus.

So far, products and exponentials compare implementations that already live in one local category. An implementation pipeline poses a different question: when the cost emitted by one runner is itself executed by another runner, which state transitions and costs result? That is the horizontal operation of A.3.

A.3: Horizontal composition and two residuals

Suppose one scheduler translates a logical tick into a variable number of intermediate ticks, while a second scheduler translates each intermediate tick into lower-level work. We want their composite implementation, not a pair of machines constrained to have the same observable behaviour. This distinction explains why there are *two* products in the story: the Cartesian product of compatible machines in A.2, and sequential or horizontal execution $\odot$ in this section. Once execution has been defined, its right adjoints answer the corresponding universal problems: what machine must be supplied before or after a fixed runner to obtain a specified result?

Execution is not synchronisation

For unary runners $F = (S, f, k)$, $G = (U, g, \ell)$,

$$G \odot F = \big(S \times U,\ (s, u) \mapsto (f(s), g^{k(s)}(u)),\ (s, u) \mapsto \textstyle\sum_{j<k(s)} \ell(g^j(u))\big). \tag{4}$$

If $K_s(n) = \sum_{i<n} k(f^i(s))$ and $H_u(n) = \sum_{j<n} \ell(g^j(u))$, then

$$K^{G\odot F}_{(s,u)} = H_u \circ K_s. \tag{5}$$

The state carrier of $G \odot F$ is the unrestricted Cartesian product, unlike the Cartesian product $G \times F$ in $\mathcal{H}$. Horizontal composition need not preserve finite products. For example, take $a = \underline{1}$ and $b = \underline{2}$ constant streams. Then $\mathsf{y}_a \times \mathsf{y}_b = 0$, whereas

$$(U_0 \odot \mathsf{y}_a) \times (U_0 \odot \mathsf{y}_b) \cong \mathsf{y}_{\underline{0}} \times \mathsf{y}_{\underline{0}} \cong \coprod_{d\in\mathbb{Z}} \mathsf{y}_{\underline{0}}.$$

Horizontal composition does preserve colimits separately, because the coalgebra forgetful functors create colimits and $- \times S$ preserves colimits in Set.

As $\mathcal{H}$ is locally presentable, both multiplications admit right adjoints:

$$G \odot - \dashv \mathsf{L}_G, \qquad - \odot F \dashv \mathsf{R}_F.$$

They must be distinguished from Cartesian exponentiation $(-)^F$. Yoneda yields

$$(\mathsf{L}_G H)(a) \cong \mathcal{H}(G \odot \mathsf{y}_a, H), \qquad (\mathsf{R}_F H)(a) \cong \mathcal{H}(\mathsf{y}_a \odot F, H).$$

Cost and time rescaling

The one-state runner U_c is a simple test of these universal questions. It implements each tick by exactly c ticks. Putting it after another runner scales the cost of each source step, while putting it before that runner groups steps into blocks. The two adjoints to these operations need not agree, and their difference is already visible in a finite collection of clock phases.

Let U_c denote the one-state runner with constant cost $c \geq 0$. Then

$$U_c \odot F = (S, f, ck), \qquad F \odot U_c = \left(S, f^c, s \mapsto \sum_{j<c} k(f^j(s))\right).$$

Thus left multiplication scales emitted cost, while right multiplication groups source ticks into blocks. For the alternating two-phase runner F from the introduction, $U_2 \odot F$ emits $0, 4, 0, 4, \ldots$, whereas $F \odot U_2$ emits 2 every tick (since $f^2 = \mathrm{Id}$). The order of execution is observably consequential. For every c, $(\mathsf{L}_{U_c} H)(a) = H(ca)$, where ca multiplies each entry of the stream by c. For $c \geq 1$,

$$\mathsf{L}_{U_c} U_d \cong \begin{cases} U_{d/c} & c \mid d, \\ 0 & c \nmid d. \end{cases}$$

For the other residual define $\beta_c(a)_n = \sum_{j<c} a_{cn+j}$. For $c \geq 1$, splitting the states of $\mathsf{y}_a \odot U_c$ according to their index modulo c gives

$$(\mathsf{R}_{U_c} H)(a) \cong \prod_{r=0}^{c-1} H(\beta_c(\sigma^r a)). \tag{6}$$

In particular $P_c := \mathsf{R}_{U_c} U_1$ has state set $\{0, \ldots, c-1\}$, successor $i \mapsto i+1 \pmod c$, and cost 0 except for cost 1 when $i = c-1$. Equivalently its states are cyclic binary cost patterns with total 1 per period. More generally $\mathsf{R}_{U_c} U_d$ consists of length-c weak compositions of d, rotated on each tick.

Remark A.3.1 (Boundary case). For $c = 0$, equation (6) does not apply. Indeed

$$\mathsf{y}_a \odot U_0 \cong \coprod_{n\in\mathbb{N}} U_0,$$

where U_0 is a *one-state, stationary* zero-cost runner and is not $\mathsf{y}_{\underline{0}}$. Thus

$$(\mathsf{R}_{U_0}H)(a) \cong \mathcal{H}(\mathsf{y}_a \odot U_0, H) \cong \mathrm{Fix}\big(h|_{H(\underline{0})}\big)^{\mathbb{N}},$$

independently of a, where h denotes the transition of H. The successor on the right shifts the sequence of fixed points. Since $U_1(\underline{0}) = \varnothing$, the earlier conclusion $\mathsf{R}_{U_0}U_1 = 0$ remains correct.

Composition by kernels

The preceding formulas concern particular machines. To see whether horizontal execution can be recovered from the local behaviour category itself, use representables as probes and determine what their composites look like. The resulting kernel assembles general runners by density. It is a bridge toward the later profunctorial view of behaviours, rather than a claim that horizontal composition is an ordinary functor-category product.

With $\mathsf{y}_a = \mathcal{B}(a, -)$, define

$$\mathsf{K}(a, b; d) = (\mathsf{y}_b \odot \mathsf{y}_a)(d).$$

The underlying states of $\mathsf{y}_b \odot \mathsf{y}_a$ are $(n, m) \in \mathbb{N}^2$; its cumulative cost at such a state is

$$D_{n,m}(r) = B\big(m + A(n + r) - A(n)\big) - B(m),$$

where $A(n) = \sum_{i<n} a_i$ and $B(n) = \sum_{j<n} b_j$. Thus $\mathsf{K}(a, b; d)$ is the subset of pairs whose cumulative cost is $\sum_{i<r} d_i$ for every r. Density of representables and separate cocontinuity give

$$(G \odot F)(d) \cong \int^{a,b} F(a) \times G(b) \times \mathsf{K}(a, b; d). \tag{7}$$

This is a Day-convolution-type kernel formula, not the ordinary pointwise Cartesian product of functors. Associativity of runner composition gives the corresponding coend coherence. The exact generalisation of such kernels to all accessible measuring comonads remains a separate question. The next question is semantic rather than purely coalgebraic: if a runner translates effects, what does that translation do to an algebra in which those effects are interpreted?

A.4: Handlers, induction, and a phase adjunction

A machine does not merely produce traces. An R-algebra is a way of *handling* residual computations, by interpreting each residual operation as an operation on values. A runner $T \rightsquigarrow R$ allows this R-handler to be used as a T-handler, now depending on the runner state. This is delegation. The adjoint induction construction reverses the corresponding universal problem. For unary effects, both sides can be written without abstraction, and the phase machines of A.3 reveal an unexpectedly concrete implementation of right Kan extension (coinduction).

For $\rho = (S, f, k)$, the delegation of an algebra (B, b), $b : B \to B$, is $(B^S, D_{\rho,b})$, with

$$(D_{\rho,b}F)(s) = b^{k(s)}F(f(s)), \qquad (D^n_{\rho,b}F)(s) = b^{K_s(n)}F(f^n(s)). \tag{8}$$

It has a left adjoint Ind_ρ. Given a unary algebra (A, a), this is the unary algebra presented by symbols $[x, s]$ and the relations

$$[a(x), s] = b^{k(s)}[x, f(s)]. \tag{9}$$

Here b denotes the formal residual unary operation; explicitly the carrier is the quotient of $\mathbb{N} \times A \times S$ by the congruence generated by $(n, a(x), s) \sim (n + k(s), x, f(s))$. The adjunction follows from the universal property of this presentation.

For U_c, delegation is restriction along $\varphi_c(n) = cn$:

$$\mathrm{Del}_{U_c}(B, b) = (B, b^c) =: \mathrm{Res}_c(B, b).$$

Proposition A.4.1. *For every $c \geq 1$, the phase runner $P_c = \mathsf{R}_{U_c} U_1$ realises coinduction:*

$$\mathrm{Ind}_{U_c} \dashv \mathrm{Res}_c \dashv \mathrm{Del}_{P_c}, \qquad \mathrm{Ind}_{P_c} \cong \mathrm{Res}_c\,.$$

On (B, b), Del_{P_c} has carrier B^c and endomorphism $\delta_b(x_0, \ldots, x_{c-1}) = (x_1, \ldots, x_{c-1}, b(x_0))$.

Proof. A map $h : (A, a^c) \to (B, b)$ is equivalent to the homomorphism $\widehat{h} : (A, a) \to (B^c, \delta_b)$ with $\widehat{h}(x) = (h(x), h(ax), \ldots, h(a^{c-1}x))$. Its inverse is projection to the first coordinate. Uniqueness of left adjoints gives the second identification. □

The counit of $\mathrm{Res}_c \dashv \mathrm{Del}_{P_c}$ is first-coordinate projection $B^c \to B$; the runner-level counit $P_c \odot U_c \to U_1$, under delegation, yields instead the *diagonal* $B \to B^c$. Their opposite directions reflect the contravariance of delegation on runner 2-cells: the image of the runner-level counit is a map into the composite delegation, whereas the handler adjunction counit maps out of it. They are not the same map. The coincidence between phase coinduction and a horizontal residual is *special*, as the next section illustrates. This raises a sharp question: which monoid interpretations have a coinduction functor representable by an ordinary stateful runner, rather than requiring a more general construction?

A.5: The freeness criterion for monoid coinduction

For a monoid interpretation $\varphi : M \to N$, restriction of algebra structure is automatic and has both adjoints. But the existence of an abstract right adjoint does *not* mean that a concrete runner can realise it: runner delegation always has a carrier of the form A^S, with a fixed set of machine states S. We therefore ask whether the representing M-set for coinduction has a basis of that form. This makes the problem an exact representability question, not an analogy between adjoint functors.

Let $\varphi : M \to N$ be a monoid morphism. Restriction of left actions $\varphi^* : N\text{-}\mathbf{Set} \to M\text{-}\mathbf{Set}$ has adjoints $\mathrm{Lan}_\varphi \dashv \varphi^* \dashv \mathrm{Ran}_\varphi$. Give N the left M-action $m \cdot n = \varphi(m)n$. Then

$$\mathrm{Ran}_\varphi(A) = \mathrm{Hom}_M({}_\varphi N, A), \qquad (n \cdot F)(x) = F(xn). \tag{10}$$

Theorem A.5.1 (Freeness criterion)**.** *The functor Ran_φ is naturally isomorphic to delegation along an ordinary writer runner $P : T_N \rightsquigarrow T_M$ if and only if the left M-set ${}_\varphi N$ is free. Equivalently, there is a set S and chosen elements $r_s \in N$ giving an equivariant bijection*

$$M \times S \xrightarrow{\sim} N, \qquad (m, s) \mapsto \varphi(m) r_s.$$

Proof. If such a basis exists, uniquely decompose $r_s n = \varphi(c(s, n)) r_{s \cdot n}$. Uniqueness gives a right N-action and an M-valued cocycle. Evaluation at r_s gives $\mathrm{Hom}_M({}_\varphi N, A) \cong A^S$, and the induced N-action agrees with delegation by the runner just constructed. Conversely, delegation along a runner with state set S has underlying carrier functor $A \mapsto A^S$, represented in M-sets by the free M-set $M \times S$. Coinduction has underlying functor represented by ${}_\varphi N$. A natural isomorphism of these functors forces ${}_\varphi N \cong M \times S$ by Yoneda. □

Some test cases are worth recording:

- $\mathbb{N} \xrightarrow{\times c} \mathbb{N}$, $c > 0$: division with remainder gives the c-phase runner. At $c = 0$ the action is not free.

- $\mathbb{N}^2 \xrightarrow{(p,q)\mapsto(cp,dq)} \mathbb{N}^2$, $c, d > 0$: the basis is a $c \times d$ rectangular phase grid.
- The free-monoid map $a \mapsto aa,\ b \mapsto bb$ admits unique prefix-code reduction; its basis consists of undecoded remainders, generally infinitely many.
- For an inclusion of groups $H \hookrightarrow G$, choice of representatives for right cosets $H\backslash G$ gives a basis, and the runner cocycle is the familiar Schreier factor $r_s g = c(s, g) r_{s\cdot g}$.
- The commutativity quotient $A^* \twoheadrightarrow \mathbb{N}^2$ is not injective and cannot yield a free action. Its right adjoint selects the largest invariant subalgebra on which the two endomorphisms commute, which may be empty even when the original carrier is nonempty.
- The inclusion $\mathbb{N} \hookrightarrow \mathbb{Z}$ has a nonfree action; its right adjoint selects bi-infinite compatible orbits, again potentially empty.

Since $N \neq \varnothing$, freeness in this theorem implies that φ is injective. Injectivity by itself is insufficient (as $\mathbb{N} \hookrightarrow \mathbb{Z}$ shows). The criterion is for *ordinary runner delegation*: more general semantic implementations may use other constructions. In the biset formulation of the next section*, the theorem is an Eilenberg–Watts-style representability statement: the restriction bimodule ${}_\varphi N_N$ must be free on its left M-side. The Yoneda proof above makes the freeness condition explicit. Thus the phase construction is the positive case of a general criterion; commutative quotients and nonfree actions show where it fails. But all of this still treats morphisms between machines as strict maps of states. To compare two implementations that differ by a residual correction, we need a broader notion of arrow.

A.6: Left-free bisets and complete handler semantics

Here the second dimension becomes unavoidable. A strict runner map changes the representation of a machine's hidden state but must preserve the emitted residual computation exactly. That is too rigid for many natural equivalences: changing a basis, shifting a clock phase, or changing representatives of a coset can require an additional residual operation. A meaningful map between implementations should therefore be permitted to *perform* a residual effect as it translates states. The guiding criterion is semantic: such a comparison ought to induce a natural transformation between the corresponding handler/delegation functors. The writer case explains the construction by elementary algebra; Yoneda then gives the general Set-monad classification.

The writer runner as a left-free biset

For monoids M, N, regard a biset as a set with commuting left M- and right N-actions. A writer runner $\rho : T_N \rightsquigarrow T_M$ with state set S, action $s \cdot n$, and cocycle $c(s, n)$ corresponds to the (M, N)-biset

$$P_\rho = M \times S, \qquad (m, s) \cdot n = (m\, c(s, n), s \cdot n),$$

whose left M-action is free on the indicated basis S. Conversely, choosing a left basis for a left-free biset yields the runner cocycle by decomposing each $(1, s) \cdot n$ in that basis. In particular, *strict* runner maps are the biset maps taking each displayed basis element to a basis element; arbitrary biset maps need not do so.

For a left M-set A and a left N-set X, the standard biset adjunction is

$$\mathrm{Hom}_N(X, \mathrm{Hom}_M(P_\rho, A)) \cong \mathrm{Hom}_M(P_\rho \otimes_N X, A). \tag{11}$$

Consequently,

$$\mathrm{Del}_\rho = \mathrm{Hom}_M(P_\rho, -), \qquad \mathrm{Ind}_\rho = P_\rho \otimes_N -.$$

Here $\mathrm{Hom}_M(P_\rho, A)$ has its left N-action by precomposition with the right N-action on P_ρ; the identification with A^S is evaluation on basis elements. The orientation is: $T_N \rightsquigarrow T_M$ induces a functor M-**Set** $\to N$-**Set**.

Proposition A.6.1 (Writer horizontal composition)**.** *For composable writer runners $P : T_M \rightsquigarrow T_N$ and $Q : T_N \rightsquigarrow T_L$, the composite corresponds to*

$$P_{Q \odot P} \cong P_Q \otimes_N P_P.$$

If $P_P \cong N \times S$ and $P_Q \cong L \times U$, there is a basis-dependent (L, M)-biset isomorphism

$$(L \times U) \otimes_N (N \times S) \cong L \times (S \times U).$$

Tensor product associativity and its unitors therefore supply the associator and unitor for writer-runner composition. The corresponding handler compositor is the tensor–Hom isomorphism

$$\mathrm{Del}_{Q \odot P} \cong \mathrm{Del}_P \circ \mathrm{Del}_Q .$$

Proof. Explicitly, with $d(t, n)$ the residual L-cocycle of Q, set

$$[(\ell, t), (n, s)] \longmapsto (\ell \, d(t, n), (s, t \cdot n)),$$

Balancing over N makes this independent of representatives. The inverse sends $(\ell, (s, t))$ to $[(\ell, t), (1_N, s)]$. The left L-actions clearly agree, and the right M-action agrees because balancing by the cocycle $c(s, m)$ reproduces the state update $(s, t) \cdot m = (s \cdot m, t \cdot c(s, m))$ and the residual effect $d(t, c(s, m))$. Thus this is an (L, M)-biset isomorphism. The coherence is inherited from associativity and units of tensor products of bisets. □

The bicategory presented by writer runners with effectful cells is *biequivalent to* the wide, locally full sub-bicategory of bisets (equivalently, profunctors between one-object categories) whose 1-cells are free as left sets. A runner makes a *choice of basis*; a left-free biset need not come with such a choice. The comparison is fully faithful on each category of 2-cells and essentially surjective on 1-cells, but is not literally injective on 1-cells. Rebasings, including gauge changes, are isomorphisms in the biset bicategory. The *effectful* 2-cells below are all biset maps, not a new ad hoc notion of cell. The ordinary hom-topos of strict runner maps remains a different local category.

The writer formula and gauge changes

Let $\rho = (S, \cdot, c)$ and $\rho' = (S', \cdot', c')$ be runners $T_N \rightsquigarrow T_M$. A *writer-effectful comparison* is

$$h : S \to S', \qquad u : S \to M$$

such that for all s, n,

$$h(s \cdot n) = h(s) \cdot' n, \qquad c(s, n)u(s \cdot n) = u(s)c'(h(s), n). \tag{12}$$

It gives, on any left M-set A, a homomorphism of delegated N-sets

$$\Theta_A : A^{S'} \to A^S, \qquad (\Theta_A F)(s) = u(s) \cdot_A F(h(s)). \tag{13}$$

The proof substitutes (12) into the action laws. Unlike a strict runner map, this can perform a state-dependent residual correction.

For a group residue, fix the state action and let $h = \mathrm{Id}$. Then $c'(s, n) = u(s)^{-1}c(s, n)u(s \cdot n)$, a cocycle gauge change. For subgroup coinduction, changing chosen coset representatives

$r'_s = v_s r_s$ yields $c'(s,g) = v_s c(s,g) v_{s\cdot g}^{-1}$, and the handler-coordinate change is $F(s) \mapsto v_s \cdot F(s)$. These are natural isomorphisms usually not induced by strict state maps.

The biset interpretation makes the comparison equation transparent: an arbitrary left-M-linear map $M \times S \to M \times S'$ is determined by $(1,s) \mapsto (u(s), h(s))$, and right N-equivariance is precisely (12). Thus the effectful hom-category is *equivalent to* the category of left-free (M,N)-bisets and all biset maps; forgetting the chosen runner bases is not an equality of categories. The correspondence between these maps and all natural handler transformations follows both from this description and from the general Yoneda argument below.

Vertical composition is

$$(j,v) \circ (h,u) = (j \circ h,\ s \mapsto u(s)v(h(s))). \tag{14}$$

A particularly useful check is the one-state identity runner for the unary monad: its effectful endomorphisms are indexed by $r \in \mathbb{N}$ and act on (B,b) by b^r. Strict runner endomorphisms see only the identity.

The Set-monad classification

The biset calculation shows that, for writers, an effectful comparison is nothing more mysterious than a map of the corresponding bimodules; it also explains why strict runner morphisms miss genuine semantic transformations. We now remove the writer hypothesis. A compatible Kleisli state translation is the correct replacement for a function $S \to S'$, and the theorem below says that *all* natural handler transformations arise in exactly this way.

Let T, R be monads on Set, and ρ, ρ' runners with respective state sets S, S'. For a Kleisli map $\alpha : S \to RS'$, define its strength extension

$$\bar{\alpha}_Z : Z \times S \to R(Z \times S'), \qquad \bar{\alpha}_Z(z,s) = R(s' \mapsto (z,s'))\alpha(s).$$

Write $p^\sharp = \mu\, Rp$ for Kleisli extension.

Theorem A.6.2 (Effectful comparison/handler theorem)**.** *The following data correspond bijectively:*

1. *Kleisli arrows $\alpha : S \to RS'$ satisfying, naturally in X,*

$$\bar{\alpha}_X^\sharp \rho_X = (\rho'_X)^\sharp \bar{\alpha}_{TX} : TX \times S \longrightarrow R(X \times S'); \tag{15}$$

2. *natural transformations $\mathrm{Del}_{\rho'} \Rightarrow \mathrm{Del}_\rho$ of functors $\mathrm{Alg}(R) \to \mathrm{Alg}(T)$.*

The associated component at the R-algebra (B,b) is

$$(\Theta_B^\alpha F)(s) = b\big(RF(\alpha(s))\big). \tag{16}$$

Strict state maps h correspond to $\alpha = \eta_R h$.

Proof. The underlying carrier functor of Del_ρ is $B \mapsto B^S$, represented on $\mathrm{Alg}(R)$ by the free algebra RS: $B^S \cong \mathrm{Alg}(R)(RS, B)$. Yoneda identifies natural carrier maps $B^{S'} \to B^S$ with algebra maps $RS \to RS'$, or Kleisli arrows $\alpha : S \to RS'$, and gives formula (16). It remains to impose preservation of the delegated T-algebra structures. Expanding their definition using ρ, monad laws and strength shows that (15) suffices. Conversely take the free R-algebra $B = R(X \times S')$ and the map $X \to B^{S'}$, $x \mapsto (s' \mapsto \eta_R(x,s'))$. Evaluating the T-homomorphism equation at this map recovers exactly (15). Thus it is also necessary. □

This theorem is a precise *local* statement (fixed source and residual monads). It can be read as Yoneda for represented lifts of comodels: the free-R-algebra embedding of the Kleisli category is fully faithful, and comparison of delegation functors recovers a unique compatible Kleisli state translation. We do not claim novelty for the statement without a closer priority check against Uustalu, Ahman–Bauer and Garner [5, 2]; our explicit proof records the precise form used here. The mate of Θ^{α} under $\mathrm{Ind}_{\rho} \dashv \mathrm{Del}_{\rho}$ and $\mathrm{Ind}_{\rho'} \dashv \mathrm{Del}_{\rho'}$ is $\mathrm{Ind}_{\rho} \Rightarrow \mathrm{Ind}_{\rho'}$. In the writer case it sends $[x, s]$ to $u(s) \cdot [x, h(s)]$.

Lemma A.6.3 (Coherent delegation under runner composition)**.** *Let $\rho : T \rightsquigarrow R$ and $\sigma : R \rightsquigarrow Q$ be Set-based stateful runners on state sets S and U. Define $\sigma \odot \rho$ by the standard strong state-passing composite*

$$TX \times S \times U \xrightarrow{\rho_X \times 1_U} R(X \times S) \times U \xrightarrow{\sigma_{X \times S}} Q((X \times S) \times U),$$

followed by reassociation. For every Q-algebra (B, b), Cartesian currying gives a natural T-algebra isomorphism

$$\kappa_{\rho,\sigma,B} : \mathrm{Del}_{\sigma \odot \rho}(B, b) \xrightarrow{\cong} \mathrm{Del}_{\rho}(\mathrm{Del}_{\sigma}(B, b)), \qquad B^{S \times U} \cong (B^{U})^{S}.$$

These isomorphisms are natural in Q-algebra maps and in strict runner 2-cells, and satisfy the unit and associativity coherence laws.

Proof. For a computation $t \in T(B^{S \times U})$, evaluation of the T-algebra structure on the left at (s, u) follows the displayed state-passing composite and then applies Q-evaluation $Q((F, s, u) \mapsto F(s, u))$, followed by b. On the right, first curry F to $s \mapsto (u \mapsto F(s, u))$. The R-algebra structure on B^{U} is delegated along σ; applying ρ and then evaluating at s reduces to the same composite by naturality of ρ in its value argument and naturality of σ with respect to $(F, s) \mapsto (u \mapsto F(s, u))$. Thus the currying bijection preserves the T-operations, not only the carrier sets. For a strict runner map, the induced transformation is precomposition with its state map. Naturality of Cartesian currying with respect to products of state maps gives naturality in these 2-cells. Both sides of the unit and pentagon diagrams send a function of three or more state arguments to the same iterated curry; the diagrams commute by the standard Cartesian associativity and unit identities. □

Horizontal pasting: what is checked

For writer runners $P : T_M \rightsquigarrow T_N$, $Q : T_N \rightsquigarrow T_L$, and comparisons $(h, u) : P \to P'$, $(j, v) : Q \to Q'$, let $d(t, n) \in L$ denote Q's residual cocycle. The horizontal paste on composite state (s, t) is

$$\begin{aligned} H(s, t) &= (h(s), j(t \cdot u(s))), \\ w(s, t) &= d(t, u(s))\, v(t \cdot u(s)). \end{aligned} \tag{17}$$

Direct substitution in the cocycle equations proves compatibility; more conceptually, this is just tensoring biset maps. There is an apparently different formula, obtained by applying the Q-cell before executing the correction $u(s)$:

$$H_{\mathrm{alt}}(s, t) = (h(s), j(t) \cdot' u(s)), \qquad w_{\mathrm{alt}}(s, t) = v(t)\, d'(j(t), u(s)).$$

It agrees with (17) because the comparison (j, v) satisfies

$$j(t \cdot u(s)) = j(t) \cdot' u(s), \quad d(t, u(s))\, v(t \cdot u(s)) = v(t)\, d'(j(t), u(s)).$$

This equality expresses the interchange of the two effectful corrections. Associativity, identities and interchange now follow from the standard bicategory of bisets, rather than requiring an independent coherence proof in the writer case.

For general Set monads, the construction follows from Lemma A.6.3 rather than from an unverified carrier isomorphism. The coherent assignment $\rho \mapsto \mathrm{Del}_\rho$, with its reversed variance on horizontal arrows and 2-cells, lands in the 2-category of algebra categories and functors. Retaining the original objects and horizontal runners, but taking as cells all natural transformations $\mathrm{Del}_{\rho'} \Rightarrow \mathrm{Del}_\rho$, inherits horizontal composition and coherence from this assignment. Theorem A.6.2 then transfers that bicategory structure to the compatible Kleisli translations. This supplies an effectful Set-runner bicategory under the stated strong state-passing composition. It is not the full double category with arbitrary tight boundary maps. Its explicit non-writer horizontal-paste formula is still worth developing.

For squares with tight monad morphisms $\lambda : T \to T'$, $\mu : R \to R'$, first transport the two runners to common boundaries T, R'. The candidate cell $\gamma : S \to R'S'$ then satisfies equation (15) for those transported runners. Establishing pseudofunctoriality with respect to *both* boundary interpretations is part of the remaining structural task.

A.7: Nondeterministic and lax comparisons

Polynomial measuring comonads have supplied particularly transparent local categories, and writer residues have supplied a particularly tractable model of 2-cells. Neither feature should be built into the definition of a runner. We therefore make a deliberate stress test: residual effects may now branch nondeterministically. This both limits the earlier topos picture and shows why exact equality of implementations is sometimes too demanding. Inclusion of sets of possible residual successors leads naturally to a lax comparison, which can be recognised directly in the order on handler semantics.

A runner $\mathbb{N}y \rightsquigarrow \mathcal{P}_\mathrm{f}$ is a transition map $\delta : S \to \mathcal{P}_\mathrm{f}(S)$, and strict runner maps are functional coalgebra morphisms. Thus

$$\mathrm{Run}(\mathbb{N}y, \mathcal{P}_\mathrm{f}) \cong \mathrm{Coalg}(\mathcal{P}_\mathrm{f}).$$

This category is locally presentable but *not cartesian closed*, and hence not a topos. Specifically, Johnstone, Power, Tsujishita, Watanabe and Worrell's Corollary 5.5 and Example 5.6 show that the coalgebra category of their Kuratowski functor $K = \mathcal{P}_\mathrm{f}$ is not cartesian closed [6]. Their general analysis also explains why much topos-like structure survives, while effectiveness of equivalence relations is an additional exactness concern.

A Kleisli comparison is a finite-valued map $\alpha : S \to \mathcal{P}_\mathrm{f}(S')$. For unary source, equation (15) is exactly

$$\bigcup_{t \in \delta(s)} \alpha(t) = \bigcup_{u \in \alpha(s)} \delta'(u). \tag{18}$$

If B is a join-semilattice with bottom (an algebra for $\mathcal{P}_\mathrm{f}$), delegation and comparison act by

$$(D_\delta F)(s) = \bigvee_{t \in \delta(s)} F(t), \qquad (\Theta^\alpha_B F)(s) = \bigvee_{u \in \alpha(s)} F(u).$$

Equation (18) is equivalent to $D_\delta \Theta^\alpha_B = \Theta^\alpha_B D_{\delta'}$ for every such B.

Order the Kleisli homs by inclusion and weaken equality to

$$\bigcup_{t \in \delta(s)} \alpha(t) \subseteq \bigcup_{u \in \alpha(s)} \delta'(u). \tag{19}$$

Then

$$D_\delta \Theta^\alpha_B \leq \Theta^\alpha_B D_{\delta'}$$

pointwise in every semilattice algebra. Conversely, take the free algebra $B = \mathcal{P}_\mathrm{f}(S')$ and $F(u) = \{u\}$ to recover (19) from this family of inequalities. Thus the *universal* lax handler inequality is equivalent to Kleisli inclusion in this case.

Finite-valued lax comparisons compose by relational/Kleisli composition; their inclusions paste by monotonicity. Writing $s \mathcal{R} u$ for $u \in \alpha(s)$, equation (19) expands to

$$s \to t \;\; \mathcal{R} \;\; v \quad \Longrightarrow \quad \exists u \, (s \mathcal{R} u \text{ and } u \to' v).$$

This is precisely the *backward-simulation transition clause* of Lynch–Vaandrager; in Hasuo's convention it is the *oplax* Kleisli-coalgebra morphism direction for a relation $\alpha : S \to \mathcal{P}_{\mathrm{f}}(S')$ [3]. The reverse inclusion adds

$$s \mathcal{R} u \to' v \quad \Longrightarrow \quad \exists t \, (s \to t \text{ and } t \mathcal{R} v).$$

Thus exact effectful comparisons satisfy both clauses. Although the union equations express pointwise existential step matching, the quantifiers differ from those of the ordinary *forward*-simulation step. There is also a crucial *totality* caveat: the constantly empty relation $\alpha(s) = \varnothing$ satisfies both the exact and lax equations. Lynch–Vaandrager backward simulation additionally requires totality (here $\alpha(s) \neq \varnothing$ at least on the relevant reachable concrete states). Image-finiteness is automatic for our $\mathcal{P}_{\mathrm{f}}$-valued comparisons. As Urabe–Hasuo explain, totality and image-finiteness are among the extra hypotheses needed for soundness of backward simulation with respect to infinitary trace inclusion [4]. Our calculation isolates transition clauses only: complete automaton simulations must also account for initial states and termination/observations. Hasuo's general treatment uses the full powerset monad, whereas our finite-valued relation is a special case.

Stocktaking: what problem Stage A has solved

Stage A began with the problem of putting stateful implementations into a compositional, semantically meaningful setting. We can now separate four answers. Behaviour categories describe *what a particular implementation can do*; horizontal execution describes *how implementations interact*; delegation and induction describe *what they do to interpretations of programs*; and effectful cells describe *when two implementations implement the same behaviour up to residual work*. The finite-powerset test confirms that the last answer still has content when deterministic/polynomial hypotheses are abandoned, although the surrounding categorical properties change.

The mathematical core

The calculations now support a coherent three-level picture:

$$\underbrace{\mathrm{Coalg}(\{T, R\})}_{\substack{\text{strict runner hom-category} \\ \text{sometimes a presheaf topos}}} \longrightarrow \underbrace{\text{horizontal execution and local residuals}}_{\text{biclosed in the unary example}}$$

$$\downarrow$$

$$\underbrace{\text{Kleisli comparisons and handler maps}}_{\text{complete local Set-monad classification}}.$$

The freeness criterion identifies the left-free-biset case in which the right Kan extension of a monoid interpretation is itself delegation along a runner. The nondeterministic example shows how exact handler homomorphisms become inequalities after ordered laxification.

Four points not to blur

1. **Two local categories.** The coalgebra category with strict state maps and the Kleisli-enlarged category with effectful comparisons share objects but not arrows. The polynomial

presheaf-topos theorem pertains to the former; it is not automatically inherited by the latter.

2. **Three adjunction mechanisms.** Cartesian exponentials, right adjoints to horizontal runner composition, and induction–delegation of algebras are different constructions. The phase-runner coincidence with coinduction is not a general identity between them.
3. **Functoriality versus coherence.** The writer-case bicategory is biequivalent to left-free bisets, inheriting interchange from tensor products. For general Set monads, Lemma A.6.3 supplies the coherent delegation compositor needed to transport the bicategory structure through the local Yoneda correspondence. What is *not* settled is the full double-category extension with tight boundary maps, its enriched analogues, and a useful explicit formula for general pasting.
4. **Backward versus forward simulation.** The chosen finite-powerset inequality is the backward-simulation *transition clause* (oplax in Hasuo's convention), not the forward clause. It does not imply totality; start-state and observation requirements and, for infinitary trace soundness, suitable totality/image-finiteness conditions remain additional hypotheses.

A deliberate stopping point

The three adjunctions encountered locally in Stage A should not be confused with the *full Sweedler adjunctions*. Cartesian exponentiation concerns the hom-topos of strict machines, horizontal residuals invert execution on one side, and induction–delegation acts on categories of algebras. The next stage asks a different, universal question: if D is a comonad of machines and T is a theory to be run, which residual theory is freely generated by their interaction, and how does this relate by adjunction to the measuring comonad $\{T, R\}$? Stage A provides explicit implementations and handler comparisons on which to test that larger structure.

The Stage A outcome is a *self-contained Set-based calculus*, not an attempt to finish every Sweedler construction. The next broad topics may be approached after an explicit account of variance for tight boundary maps, a literature-priority check on the Yoneda formulation of Theorem A.6.2, and an exposition of the ambient hypotheses:

- Sweedler powers, copowers, measuring hom composition and their universal examples;
- order-enriched and genuinely enriched/lax versions of effectful comparisons;
- modules/comodules, parametrised handlers and the behaviour/profunctor kernel construction;
- comparison of these structures under change of ambient duoidal category.

The strongest conceptual formulation to take forward is that *runners implement theories, their cells compare implementations, and the algebra/handler semantics transports those comparisons.* Stage A supplies a precise Set-based baseline, with a clear boundary between established calculations and a future general theory. We should therefore leave the proved Stage A calculations in place while developing the full Sweedler adjunctions as the next layer, rather than treating those adjunctions as something already proved by the local closed structures above.

References

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

[2] R. Garner, *The costructure–cosemantics adjunction for comodels for computational effects*, *Mathematical Structures in Computer Science* **32**(4), 374–419 (2022). doi:10.1017/S0960129521000219.

[3] I. Hasuo, *Generic Forward and Backward Simulations*, in *CONCUR 2006*, LNCS 4137, pp. 406–420, Springer, 2006. doi:10.1007/11817949_27.

[4] N. Urabe and I. Hasuo, *Coalgebraic Infinite Traces and Kleisli Simulations*, *Logical Methods in Computer Science* **14**(3:15), pp. 1–38 (2018). doi:10.23638/LMCS-14(3:15)2018.

[5] T. Uustalu, *Stateful Runners of Effectful Computations*, *Electronic Notes in Theoretical Computer Science* **319**, 403–421 (2015). doi:10.1016/j.entcs.2015.12.024.

[6] P. T. Johnstone, J. Power, T. Tsujishita, H. Watanabe and J. Worrell, *On the structure of categories of coalgebras*, *Theoretical Computer Science* **260**, 87–117 (2001). doi:10.1016/S0304-3975(00)00124-9.

[7] I. López Franco and C. Vasilakopoulou, *Duoidal categories, measuring comonoids and enrichment*, arXiv:2005.01340. https://arxiv.org/abs/2005.01340.