Residual Sweedler Runners over Contexts

Part D — the coend classifier, restriction, scopes, and threads

Dependent Sweedler Runners sequence
Revised draft, 19 November 2026

Abstract

Part C paired covariant syntax with contravariant states and classified pure contextual runners for a unary theory. We identify the pure runners as contextual comodels and prove a two-variable adjunction for the coend pairing, using accessible syntax and residual monads and accessible state comonads. Equivalence relations on finite contexts give a runner for Staton's restriction and equality operations. A scoped nondeterminism example supplies a test for delimitation. For dynamic threads, the identity residual forces empty carriers, while a multiset residual counts actions and returns. On the one-state carrier, every residual runner identifies some causally distinct programs. A runner that stores remaining delays for the live thread IDs distinguishes both the sequential/concurrent pair and the choice of which of two children to wait for.

1 What an execution can leave behind

In Part C [1], a program marks names and introduces fresh ones. If it starts with state s at context w and ends by returning x with state s' at context w', the answer is the coend class $[x, s']$. This allows the world of the return value to differ from the world at which the program started. It also assumes that there is a return value. In the thread calculus of Kammar and colleagues [10], a program can stop without returning, or fork into threads that make separate contributions. The codomain must be able to express these outcomes if the runner is to account for them.

Let $\mathcal{W}$ be a small category of contexts, $\mathcal{P} = [\mathcal{W}, \mathbf{Set}]$ the category of continuation functors, and $\mathcal{S} = [\mathcal{W}^{\mathrm{op}}, \mathbf{Set}]$ the category of state presheaves. For $X \in \mathcal{P}$ and $S \in \mathcal{S}$, write

$$X \otimes S = \int^{w \in \mathcal{W}} X(w) \times S(w).$$

For $f : w \to v$, the relation identifies $[X(f)x, s]$ with $[x, S(f)s]$. Renaming the program and restricting the state are two ways of making the same pair. The pairing preserves colimits separately in each variable.

Fix a monad T on $\mathcal{P}$ and a monad r on $\mathbf{Set}$. The syntax monad governs the program that is supplied. The residual monad governs the outcomes the runner returns and how those outcomes combine under continuation substitution. Writing $G_S(X) = X \otimes S$, we make this precise as follows.

Definition 1.1. A *residual coend runner* on S is a natural transformation

$$\alpha_X : G_S(TX) \longrightarrow r(G_S X)$$

satisfying

$$\begin{aligned} \alpha_X \, G_S(\eta^T_X) &= \eta^r_{G_S X}, \\ \alpha_X \, G_S(\mu^T_X) &= \mu^r_{G_S X} \, r(\alpha_X) \, \alpha_{TX}. \end{aligned} \tag{1}$$

A runner morphism $f : (S, \alpha) \to (S', \alpha')$ is a presheaf map with $r(X \otimes f)\alpha_X = \alpha'_X(TX \otimes f)$.

The right side of (1) runs the program with its continuations still suspended, then runs each supplied continuation as prescribed by the residual. The two executions agree with substituting first and running the resulting program.

1.1 Parameterized theories and their comodels

Parameterized algebraic theories specify the parameters that a computation can use and the parameters bound for each of its continuations. Write $y_w = \mathcal{W}(w, -)$. A term at parameter context p with computation variables of parameter arities $m_1, \ldots, m_k$ corresponds to a Kleisli map

$$t : y_p \longrightarrow T\left(\coprod_i y_{m_i}\right).$$

Unit inserts computation variables, and multiplication substitutes computations for them. Staton's abstract clones on the finite coproducts of representables correspond to cartesian-strong monads on $[\mathcal{W}, \mathbf{Set}]$ preserving sifted colimits [4, Proposition 17]. This specifies the underlying accessible syntax monad needed here as well as its strength. The ordered linear theory of scopes instead uses strength for Day convolution [8, Proposition 4.6].

For $r = \mathrm{Id}$, the coend runner assigns to t a cooperation

$$S(p) \;\cong\; G_S(y_p) \;\longrightarrow\; G_S\left(T\left(\coprod_i y_{m_i}\right)\right) \;\longrightarrow\; G_S\left(\coprod_i y_{m_i}\right) \;\cong\; \coprod_i S(m_i).$$

The first and last isomorphisms are co-Yoneda and preservation of coproducts. The cooperation chooses a continuation and hands it a state in the appropriate context. In the ordinary one-context case it has the familiar type $S \to \coprod_i S$. In the restriction example below it can supply a state in a larger context than the one in which execution started.

For a sifted-colimit-preserving T, this identifies the pure coend runners with finite-coproduct-preserving functors from the Kleisli category restricted to finite coproducts of representables to $\mathbf{Set}$. In one direction the runner extends G_S along the Kleisli inclusion by its return and substitution laws. Conversely, restriction to the pure maps between representables defines S, and coproduct preservation identifies the functor on every arity with G_S. Its actions on the restricted Kleisli maps determine α on the arities, using $G_S(TB) = \int^w (TB)(w) \times S(w)$ and compatibility with the pure maps. Sifted-colimit density extends α to all X and extends the unit and substitution checks from the arities. In the substitution check, Kleisli composites with intermediate objects among the arities suffice, since TB itself is a sifted colimit of arities. This gives an ordinary categorical comodel description of the underlying monad.

Møgelberg and Staton's *Linear Usage of State* gives the published state-access form $B \cdot S \to \bigoplus_i (A_i \cdot S)$ for a parameter of type B and continuation arities A_i, with $\cdot$ the copower in their enriched state category [5, Theorem 8.1]. Staton relates their effect theories and comodels explicitly to the PAT presentation [6]. The coend cooperation above uses the same arity pattern with G_S as the pairing functor. The constructions below use the underlying monads. They leave compatibility with the chosen strength as a further condition.

2 The measuring comonad

2.1 Cofree runners

Theorem 2.1. *If T and r are accessible, the forgetful functor $U : \mathrm{Run}_{\mathcal{W}}(T, r) \to \mathcal{S}$ is comonadic. Its right adjoint gives a cofree runner on each presheaf Y. Write $C_{\mathcal{W}}(T, r)$ for the induced comonad,*

so

$$\mathrm{Run}_{\mathcal{W}}(T, r) \simeq \mathrm{Coalg}(C_{\mathcal{W}}(T, r))$$

over $\mathcal{S}$.

Proof. By uniformization [2, Theorem 2.19], choose a regular λ such that $\mathcal{P}$ is locally λ-presentable, both monads preserve λ-filtered colimits, and T preserves λ-presentable objects. For fixed S, the two functors in α preserve these colimits in X. A natural transformation between them is therefore specified by its components and naturality on a small full category of λ-presentable objects. Choose this category closed under T by adjoining the finite T-iterates of a small set of representatives.

In the carrier variable S, the component data form the inserter of the accessible functors with components $S \mapsto TX \otimes S$ and $S \mapsto r(X \otimes S)$ in a set-indexed product of copies of **Set**. Naturality is imposed by equifiers. Impose the unit and multiplication equations at the same presentable X's. Since TX is among the test objects, α_{TX} is a component of the restricted family, and the equations are again equifiers. Both sides of each law are natural in X between λ-accessible functors, so the restricted tests suffice. The accessibility of inserters and equifiers gives an accessible category of runners and an accessible forgetful functor [2].

For a diagram of runners with carrier colimit S, the maps into $r(X \otimes S)$ induced by its cocone define α_X uniquely, since $TX \otimes -$ preserves the carrier colimit. Naturality and the laws follow by precomposing with the corresponding colimit injections, including those of $T^2X \otimes S$ for the multiplication law. Thus U creates colimits. The runner category is locally presentable, and the colimit-preserving functor U has a right adjoint. For U-split equalizers the functors $TX \otimes -$ and $r(X \otimes -)$ preserve the underlying split equalizer. The structure therefore restricts uniquely, and the equations follow by postcomposing into the original runner. Since U also reflects isomorphisms, the dual Beck theorem applies. □

2.2 The universal interaction

For a comonad D on $\mathcal{S}$, an interaction is natural in X, Y and has type

$$\theta_{X,Y} : TX \otimes DY \longrightarrow r(X \otimes Y).$$

The unit law is $\theta_{X,Y}(\eta^T_X \otimes DY) = \eta^r_{X \otimes Y}(X \otimes \varepsilon_Y)$. The multiplication law compares $\theta_{X,Y}(\mu^T_X \otimes DY)$ with the composite

$$T^2X \otimes DY \xrightarrow{1 \otimes \delta_Y} T^2X \otimes D^2Y \xrightarrow{\theta_{TX,DY}} r(TX \otimes DY)$$
$$\xrightarrow{r(\theta_{X,Y})} r^2(X \otimes Y) \xrightarrow{\mu^r} r(X \otimes Y).$$

Here ε and δ are the counit and comultiplication of D.

Theorem 2.2. *For T, r as in Theorem 2.1, there is a natural bijection*

$$\{\textit{interactions of } T, D \textit{ into } r\} \cong \mathrm{Comon}(\mathcal{S})\big(D, C_{\mathcal{W}}(T, r)\big).$$

Proof. Given θ and a D-coalgebra $\gamma : S \to DS$, put $\alpha^\gamma_X = \theta_{X,S}(1 \otimes \gamma)$. The unit law uses $\varepsilon_S \gamma = 1$. For multiplication, insert $\delta_S \gamma = D\gamma\, \gamma$ in the interaction law and use naturality in Y along γ. The resulting composite is the right side of (1). A D-coalgebra map gives a runner morphism by naturality. This defines a functor from $\mathrm{Coalg}(D)$ to the runner category over $\mathcal{S}$.

Conversely, apply such a functor to the cofree coalgebra (DY, δ_Y), with resulting runner α^{DY}. Define $\theta_{X,Y} = r(X \otimes \varepsilon_Y)\alpha^{DY}_X$. The counit and runner unit give the interaction unit. The runner

morphism $\delta_Y : DY \to D^2Y$ and $\varepsilon_{DY}\delta_Y = 1$ allow $\theta_{TX,DY}(1 \otimes \delta_Y)$ to be rewritten as α_{TX}^{DY} in the multiplication law. The runner multiplication law and naturality of μ^r then yield the interaction multiplication law. These constructions are inverse. In one direction use the counit identity and naturality of θ. In the other use that γ is a coalgebra morphism from (S, γ) to (DS, δ_S), followed by $\varepsilon_S\gamma = 1$. Finally, by Theorem 2.1, functors over $\mathcal{S}$ from D-coalgebras to runners correspond to comonad morphisms into $C_{\mathcal{W}}(T, r)$. □

For finite injections $\mathbf{I}$ and the unary history monad T_{hist} of Part C, the representing universal property gives

$$C_{\mathbf{I}}(T_{\text{hist}}, \text{Id})Y(n) \;\cong\; (C_{\text{hist}}Y)(n) \;=\; \prod_{h \in \text{Hist}(n)} Y(\text{end}(h)).$$

Here h is a legal command history starting at n. The counit evaluates at the empty history and comultiplication evaluates by concatenating histories. This existence argument works directly with an accessible quotient monad, so equations of the source are already included in the input to the runner. An explicit account of the resulting cofree behaviours can still require a separate calculation. When $\mathcal{W}$ is terminal, the pairing is the product of sets and this is the usual set-based runner type. The proof adapts the accessibility argument of McDermott–Rivas–Uustalu [3, Theorem 1] to a pairing with two different variances. It concerns the underlying accessible monads and the displayed laws. A theory of interactions preserving additional strengths or with arbitrary context-indexed residuals would require corresponding extra data and laws.

2.3 The two-variable adjunction for the coend pairing

Write $K(X, Y) = X \otimes Y$ and Mon for the category of monads on the indicated base. Theorem 2.2 represents interactions by varying D. One can also vary the source T or the residual r.

Theorem 2.3. *For accessible T, D, r, with respective bases $\mathcal{P}, \mathcal{S}, \mathbf{Set}$, there are natural bijections*

$$\begin{aligned} \text{Mon}(\mathbf{Set})(D \triangleright_K T, r) &\cong \text{Mon}(\mathcal{P})(T, [D, r]_K) \\ &\cong \text{Comon}(\mathcal{S})(D, C_{\mathcal{W}}(T, r)). \end{aligned}$$

All three representing objects are accessible. Each hom-set classifies the same interactions $\theta : K(T-, D-) \Rightarrow rK(-,-)$.

Proof. The comonad and its universal property are those of Theorems 2.1 and 2.2. The cofree right adjoint is accessible because it is a right adjoint between locally presentable categories [2, Proposition 2.23], so the induced comonad is accessible.

Fix D, r. Define the functor $H = [D, r]_K$ on $\mathcal{P}$ by

$$(HX)(w) = \text{Nat}_{Y \in \mathcal{S}}\big(DY(w), rK(X, Y)\big).$$

Here natural transformations are between the two functors of Y. The covariance in w comes from precomposition with the restrictions of DY. Coend evaluation gives a natural map $e_{X,Y} : K(HX, DY) \to rK(X, Y)$. For any endofunctor F on $\mathcal{P}$, a natural map $K(FX, DY) \to rK(X, Y)$ is uniquely the evaluation of a map $F \to H$.

The unit of H transposes the counit of D followed by the unit of r. Multiplication transposes the composite

$$\begin{aligned} K(H^2X, DY) &\xrightarrow{K(1,\delta_Y)} K(H^2X, D^2Y) \xrightarrow{e_{HX,DY}} rK(HX, DY) \\ &\xrightarrow{r(e_{X,Y})} r^2K(X, Y) \xrightarrow{\mu^r} rK(X, Y). \end{aligned}$$

Transposing the unit laws reduces them to the counit and unit identities. In associativity, both composites apply evaluation three times, with iterated comultiplication of D and iterated multiplication of r. Naturality of evaluation, coassociativity and associativity identify the composites. With these structures the transpose of θ is a monad map exactly when θ satisfies the two interaction laws. To see accessibility, choose a common rank λ for D, r and restrict the naturality end to a small full category of λ-presentable Y's. Both sides of the displayed naturality formula preserve λ-filtered colimits in Y. The end is then a small limit of sets of functions with fixed domains $DY(w)$. A regular rank larger than the sizes of all these domains and the end diagrams, uniformly over the small $\mathcal{W}$, makes it commute with filtered colimits of that rank in X.

Fix D, T. For each pair X, Y, write $B = K(X, Y)$. For each $u \in K(TX, DY)$, introduce an operation of arity B. On a set A, these operations amount to maps

$$\sigma_{X,Y} : A^{K(X,Y)} \longrightarrow A^{K(TX,DY)}.$$

Impose naturality in both test objects (contravariantly after exponentiating), and impose the equations

$$\begin{aligned} \sigma_{X,Y}(k)\, K(\eta^T_X, 1) &= k\, K(1, \varepsilon_Y), \\ \sigma_{X,Y}(k)\, K(\mu^T_X, 1) &= \sigma_{TX,DY}\big(\sigma_{X,Y}(k)\big)\, K(1, \delta_Y). \end{aligned} \tag{2}$$

These are ordinary equations between terms in variables indexed by $K(X, Y)$. Indeed, k names the variables, and the right side of (2) substitutes the interpretations of the inner programs into the outer operation.

This is a small infinitary presentation. Choose common presentability ranks as in Theorem 2.1, now for T and D, and take small full categories of presentable test objects closed under their respective iterates. Each of the functors of X, Y inside the exponents preserves filtered colimits of the chosen rank separately. Exponentiating into A sends these colimits to limits, so the operations and their naturality and equations extend uniquely from the tests. The resulting set of operation arities is bounded. The free-algebra monad of the presentation is therefore accessible.

For an interaction into r and an r-algebra $a : rA \to A$, interpret the operations by $\sigma(k) = a\, r(k)\, \theta$. The equations follow from the interaction and algebra laws. Conversely, a monad map from the presented monad to r interprets the operations on each r-algebra, naturally in its homomorphisms. Evaluate $\sigma_{X,Y}$ in the free r-algebra on B at $k = \eta^r_B$ to recover $\theta_{X,Y}$. Every other assignment into an r-algebra factors through this one by a unique algebra map. Thus the equations, evaluated at the generic assignment, recover the two interaction laws. These constructions are inverse and natural, giving the copower and all the indicated bijections. □

A copower calculation. Take two discrete contexts $0, 1$ and the free monad on a single command from 0 to 1, with no command from 1. Thus $(TX)(0) = X(0) \sqcup X(1)$ and $(TX)(1) = X(1)$. Set $D = \mathrm{Id}_{\mathcal{S}}$. The returns in θ are determined by its unit law. The remaining summand is $X(1) \times Y(0)$. Its generic input uses $X = y_1$ and $Y = \mathcal{W}(-, 0)$, for which $K(X, Y) = \varnothing$. Naturality therefore makes the command's interpretation equivalent to choosing a point of $r(\varnothing)$. The multiplication law introduces no further equation because the command cannot follow itself. Thus $\mathrm{Id}_{\mathcal{S}} \triangleright_K T$ is the exception monad $1 + (-)$. A command whose target context lacks a state in the generic test becomes a nullary residual operation.

When $\mathcal{W}$ is terminal the coend is cartesian product and these universal properties recover the ordinary Set-based operations. For nonterminal $\mathcal{W}$ the syntax, state and residual bases can differ. McDermott–Rivas–Uustalu's enriched construction relates different categories by actions and copowers [3, Section 6]. Theorem 2.3 gives the corresponding representability calculation directly for a Set-valued pairing.

2.4 Successive residual runners

A runner can pass its residual to another runner. Let $H_A(Z) = Z \times A$ for the state set A of a runner from r to a second Set-monad q. Given $\alpha : G_S T \Rightarrow r G_S$ and $\beta : H_A r \Rightarrow q H_A$, the sequential interaction is the composite

$$H_A G_S T \overset{H_A \alpha}{\Longrightarrow} H_A r G_S \overset{\beta G_S}{\Longrightarrow} q H_A G_S.$$

Naturality pastes the two return and substitution laws. Since $H_A G_S \cong G_{S \times A}$, the resulting state presheaf has fibres $S(w) \times A$.

The calculation also permits intermediate theories on a different context category. A carrier $M : \mathcal{W}^{\mathrm{op}} \times \mathcal{V} \to \mathbf{Set}$ gives the pairing $G_M(X)(v) = \int^w X(w) \times M(w, v)$. A further carrier $N : \mathcal{V}^{\mathrm{op}} \times \mathcal{U} \to \mathbf{Set}$ gives $G_N G_M \cong G_{N \odot M}$, with

$$(N \odot M)(w, u) = \int^v M(w, v) \times N(v, u).$$

This is ordinary profunctor composition by the coend Fubini law [7, Section 4.2]. The same pasting works with an intermediate monad on $[\mathcal{V}, \mathbf{Set}]$. Theorem 2.3 treats the Set-valued case of the pairing.

3 Restriction and equality over finite sets

In Staton's theory [4, Section V], restriction introduces a new instance of the abstract parameter type. Equality has two continuations and obeys a substitution law in its equal branch. For the single parameter type with no additional parameter operations, take $\mathcal{W} = \mathbf{FinSet}$ on a small skeleton. Substitution can identify two names, so a test based solely on whether two symbols are literally the same would not commute with every context restriction.

There is a pure runner on the presheaf $S(w)$ of equivalence relations on w. Reindexing along $f : w \to v$ pulls back an equivalence relation on v. At $E \in S(w)$, testing $a = b$ selects the equal branch precisely when $a \; E \; b$ and leaves E unchanged. Restriction extends E by putting the fresh name in a singleton class. Pullback preserves the verdict of an equality test, and pullback along $f + 1$ commutes with adding a singleton class. Hence execution descends through the coend relation. Starting at the empty context, execution itself reaches only discrete relations: a test leaves the relation unchanged and allocation adds a singleton. Discrete relations are not closed under reindexing. Pulling back the singleton relation along a map identifying two names makes them equivalent. In fact every equivalence relation E on w is the pullback of the discrete relation on the quotient w/E. Thus admitting the reachable discrete relations and closing under reindexing forces all the states used here.

Here are the equation checks. Reflexivity and repeated tests are immediate from the verdict, and tests commute because neither changes the state. In the equal branch of the substitution law, the two maps that substitute a and b for the formal parameter are pointwise E-equivalent. Pulling back E along them produces the same state, so the coend compares the two occurrences of the continuation on equal inputs. Forgetting a newly added singleton recovers E, and exchanging two new singleton classes leaves the extension invariant under reindexing. Lastly, restriction preserves verdicts on old names and makes a comparison of a new name with an old one false. These checks hold at every state. Induction on terms gives compatibility with return and substitution and validates the restriction-and-equality equations. Thus Theorem 2.1 classifies this runner alongside all others for the same source, without requiring a product-store representation of state.

4 A substitution test for scoped nondeterminism

Matache and colleagues encode scoped effects using an ordered list of linear parameters [8, Sections 3–4]. A scoped operation opens a scope, and close consumes the innermost scope. We take the discrete category $\mathbb{N}$ of depths, using the bracketing-level families of [8, Section 2.2]. Following the first-success operation of Wu, Schrijvers and Hinze [9], consider once applied to an ordered choice between returning 1 and returning 3. Applying it and then substituting a continuation k with $k(1)$ failing and $k(3)$ returning u yields failure. If we move k inside the extent of once, the first branch fails and the second succeeds with u. At the level of the list observer, taking the first element (or the empty list) sends $[1, 3]$ to $[1]$. Taking the first element after substituting the lists $[]$ and $[u]$ instead yields $[u]$.

This is a concrete test for an interpretation of the scoped theory. It must respect substitution at the point indicated by close. The example also explains why treating the first-success operation on lists as an ordinary algebraic operation would fail. Theorem 2.1 gives an existence result for the classifier on depth-indexed state families given an accessible residual. The example provides a test for a proposed runner. A residual that realizes the full scope discipline, including failure, remains to be exhibited here.

5 Dynamic threads

5.1 The identity residual and nullary operations

Proposition 5.1. *For any context category and source monad T, an identity-residual runner has $S(w) = \varnothing$ wherever $(T0)(w) \neq \varnothing$, with 0 the initial continuation functor. If this condition holds at every context, the only carrier is the empty presheaf and the classifier is the constant-empty comonad.*

Proof. If $s \in S(w)$ and $t \in (T0)(w)$, then $[t, s] \in T0 \otimes S$. The component α_0 would have to send this element to $0 \otimes S = \varnothing$. The unique runner on the empty presheaf is cofree on every Y when the condition holds everywhere. □

A nullary operation available in every context supplies the hypothesis. For dynamic threads, take the monad T on $[\mathbf{FinRel}, \mathbf{Set}]$ of Kammar and colleagues [10, Sections 4 and 7.1]. Compound thread IDs form a join-semilattice, with 0 the empty dependency. The operation $\mathsf{fork}(a.p, q)$ starts a child q and makes its ID a available to the parent continuation p. The operation $\mathsf{wait}(a, p)$ delays p until the designated dependencies complete. The nullary operations stop and act_σ return no continuation. In particular, stop gives the obstruction above at every context. The paper uses the strength of T to interpret the programming language. The classifier here uses the underlying monad, and its runner laws impose no compatibility with that strength.

5.2 An action-counting runner

Let Σ be the set of action labels. Let $r(A)$ be the finite multisets on $\Sigma \sqcup A$. Its unit is a singleton return. Its multiplication keeps action labels and adds the multisets at each returned-value occurrence, counting multiplicities. Use the terminal state presheaf 1. Since the empty context $\emptyset$ is terminal in **FinRel**,

$$X \otimes 1 \cong \operatorname{colim}_{\mathbf{FinRel}} X \cong X(\emptyset).$$

Evaluate any return at context w by mapping its label in $X(w)$ along the unique relation $w \to \emptyset$ and taking a singleton. Evaluate fork by summing the two multiset evaluations, wait by taking the

evaluation of its continuation, stop by the empty multiset, and act_σ by the singleton action label. Reindexing is invisible to this evaluation, because all reindexings followed by the unique relation to $\emptyset$ give the same result.

To check the thread equations [10, Figure 5], the wait equations become identities after erasing wait. The law distributing wait across fork does also. Fork commutativity and associativity reduce to the corresponding laws for multiset addition. Equation (16) removes a parent that only waits for its child and promotes the child to the main thread. Equation (17) removes a child that only waits. Both removed branches contribute the empty multiset. In (17), replacing the fresh ID by the indicated dependency changes no evaluation, by reindexing invariance. The return law holds by construction. For substitution, reindexing invariance lets each suspended continuation be evaluated after reindexing to $\emptyset$. Induction on the outer term then gives (1): suspension contributes a singleton occurrence, fork adds, and the nullary actions survive multiplication as action labels. This yields a nonempty runner for the quotient theory. Its residual records all action occurrences and all returns. It identifies independent actions with ordered ones when their counts agree.

5.3 What is lost at a continuation hole

Proposition 5.2. *Suppose $\mathcal{W}$ has terminal object e, and let $c_X : X \to \Delta X(e)$ be the reindexing map to the constant functor. Every runner on the terminal presheaf 1 satisfies*

$$\alpha_X = \alpha_{\Delta X(e)} \, (T(c_X) \otimes 1)$$

under the identifications $X \otimes 1 \cong X(e) \cong \Delta X(e) \otimes 1$.

Proof. Naturality gives $r(c_X \otimes 1)\alpha_X = \alpha_{\Delta X(e)}(T(c_X) \otimes 1)$. Under coend evaluation at e, the map $c_X \otimes 1$ is $(c_X)_e = 1_{X(e)}$. □

Proposition 5.3. *Suppose Σ contains an action label. No runner on the terminal state presheaf 1 can distinguish all the closed labelled posets of the dynamic-thread theory.*

Proof. Take $e = \emptyset$ in Proposition 5.2. Let A and B be atomic actions, possibly of the same label. Form two terms in $(T^2 0)(\emptyset)$ with outer fork and child A. In the parent return a suspended computation in context $\{a\}$, where a is the child's ID. Choose the suspended computation to be $q = \mathsf{wait}(a, B)$ in one term and $q' = B$ in the other. They have the same image under $(T0)(\{a\} \to \emptyset)$ because reindexing sends a to the empty dependency and $\mathsf{wait}(0, B) = B$.

Thus $T(c_{T0})$ identifies the two outer terms. Proposition 5.2 makes their images under α_{T0} equal, and (1) gives equal residuals after substitution. The fully substituted terms are the sequential and concurrent programs t_1 and t_2 of [10, Example 4.1]. One gives two ordered occurrences and the other gives two incomparable occurrences, so their labelled posets differ [10, Theorem 5.4]. □

5.4 A runner that retains thread dependencies

The terminal carrier is essential to Proposition 5.3. We can distinguish the same programs using a monad on sets if the state paired with a continuation retains a separate delay for each thread ID. The quantity observed below is the completion time of the main thread.

Use the presheaf $S(w) = \mathbb{N}^w$. A state s assigns a remaining delay to each name. An ID $a \subseteq w$ has delay $d_a(s) = \max_{i \in a} s_i$, with empty maximum 0. A relation $f : w \to v$ restricts $s \in S(v)$ by $(S(f)s)_i = d_{f(i)}(s)$, where $f(i) = \{j \in v : i \, f \, j\}$. Composition uses the associativity of maximum. Write $s \ominus d$ for the state obtained by subtracting d from every delay, truncating at zero. This operation commutes with reindexing, since truncation commutes with finite maxima.

For a set A, use the residual

$$r_{\omega_1}(A) = \{F : (A \to \mathbb{N}) \to \mathbb{N} \mid F \text{ depends on at most countably many coordinates of } A\}.$$

This is a submonad of the continuation monad. The unit evaluates a test at its given input. For multiplication, if F is supported on countably many inner continuations, union their supports to get a countable support for its flattening. This monad preserves ω_1-filtered colimits. Each countable support can be lifted to one stage of the diagram, and equality of two images can be tested at one stage by lifting their joint support and realizing all the identifications between those countably many coordinates.

A test $k : X \otimes S \to \mathbb{N}$ supplies a remaining time for each returned value paired with its current state. For a program t and state s, let $E_k(t, s)$ be its main-thread completion time relative to the current instant. Interpret the operations by

$$\begin{aligned} E_k(\eta x, s) &= k([x, s]), \\ E_k(\mathsf{stop}, s) &= 0, \qquad E_k(\mathsf{act}_\sigma, s) = 1, \\ E_k(\mathsf{wait}(a, p), s) &= d_a(s) + E_k(p, s \ominus d_a(s)), \\ E_k(\mathsf{fork}(a.p, q), s) &= E_k(p, (s, E_k(q, s))). \end{aligned}$$

In the fork clause the child starts at the same instant as the parent. Its completion time becomes the new entry for a. The fork reports the completion of the parent. A later wait is what makes the child delay affect that completion.

These clauses commute with reindexing. For a return this is the coend relation. For wait it follows from $d_a(S(f)s) = d_{f(a)}(s)$ and commutation with truncated subtraction. For fork it follows by reindexing along $f + 1$, which retains the new entry and restricts the old ones.

The thread equations. Waiting for the empty dependency costs zero. Waiting for a and then for b takes $\max(d_a, d_b)$ and shifts the state by that amount. After waiting for a, the remaining delays of b and $a \oplus b$ coincide. These are equations (10)–(12) of Figure 5. For (13), writing $d = d_b(s)$, waiting first and then forking runs the child from $s \ominus d$ and passes its delay q' to the parent. Distributing the wait into both branches gives the child delay $d + q'$ in the original state, and the parent's wait subtracts d, giving the same entry q' and the same parent state. In (14), each independent child is evaluated in the original state, so either order of fork delivers the same two entries to the parent under the indicated exchange of bound names. In (15), evaluate the child z first, then y with the delay of z, then x with the delay of y. Both sides give that evaluation, since x has no entry for the ID of z. In (16), the parent waits for the child's delay and stops, reporting exactly that delay. In (17), the child that waits for b and stops supplies $d_b(s)$ to the parent under the new name, just as substituting b does [10, Figure 5].

Consequently $[t, s] \mapsto (k \mapsto E_k(t, s))$ descends to $TX \otimes S$ and is natural in X. Each term has finitely many return occurrences, and the states at each occurrence form a countable set. The displayed map therefore lands in $r_{\omega_1}(X \otimes S)$. Return evaluates the supplied test. For substitution, interpret a suspended computation u paired with its current state s' by $E_k(u, s')$. Induction on the outer term gives the same evaluation as substituting before execution. In particular, in the fork clause the child has the same substituted delay on both sides, so the induction hypothesis is applied to the parent with the same extended state. This is precisely (1) for the continuation monad, and hence for its submonad r_{ω_1}.

The one- and two-child tests. There is a unique initial state at the empty context. The closed sequential and concurrent programs of Example 4.1 have observed durations 2 and 1, respectively. For the two-child test, take independent children a and b of durations 1 and 2, using an atomic action and a sequential pair of actions. In the parent, substitute either $\mathsf{wait}(a, B)$ or $\mathsf{wait}(b, B)$ at the hole in context $\{a, b\}$, with B atomic. At that hole the paired state has entries $(1, 2)$. The two programs report 2 and 3. This separates the named dependencies using a set-valued residual. Each return supplies a single coend pair, whose state records the remaining delay under each name. The residual can query different pairs as the computed child delays vary.

This duration interpretation can identify different labelled posets with the same main-thread completion time. Recovering the full causal order under substitution remains a separate problem. The labelled posets with holes from [10, Theorem 5.4] give a reference semantics for it.

References

[1] *Contextual Sweedler Runners*. Part C of this sequence.

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

[3] D. McDermott, E. Rivas, and T. Uustalu. Sweedler theory of monads. *FoSSaCS*, LNCS 13242, pp. 428–448, 2022. Author's copy.

[4] S. Staton. Instances of computational effects: an algebraic perspective. *LICS*, 2013. Author's copy.

[5] R. E. Møgelberg and S. Staton. Linear usage of state. *Logical Methods in Computer Science* 10(1:17), 2014. DOI.

[6] S. Staton. Freyd categories are enriched Lawvere theories. *Electronic Notes in Theoretical Computer Science* 303, pp. 197–206, 2014. DOI.

[7] G. L. Cattani. *Presheaf Models for Concurrency*. PhD thesis, University of Aarhus, 1999. Text.

[8] C. Matache, S. Lindley, S. Moss, S. Staton, N. Wu, and Z. Yang. Scoped effects as parameterized algebraic theories. Extended version, 2024. arXiv:2402.03103.

[9] N. Wu, T. Schrijvers, and R. Hinze. Effect handlers in scope. *Haskell Symposium*, 2014. Author's copy.

[10] O. Kammar, J. Liell-Cock, S. Lindley, C. Matache, and S. Staton. An equational axiomatization of dynamic threads via algebraic effects: presheaves on finite relations, labelled posets, and parameterized algebraic theories. *Proc. ACM Program. Lang.* 10 (POPL), pp. 1847–1875, 2026. arXiv:2602.05850v1, 5 February 2026.