Concurrent theories and their runners

An interlude between Parts VI and VII of the Sweedler runners tutorial

Draft, October 2026

Abstract

Ordinary monads explain sequential substitution, but do not by themselves distinguish parallel execution from either sequential ordering. Ordered concurrent monads add a parallel multiplication and directed interchange; concurrent pseudomonads promote the comparisons to coherent 2-cells. We develop these structures through duration grades, shuffles, and shared-state resumptions. Two explicit counterexamples show why sequential theory interpretations and isolated stateful running need not preserve concurrent behaviour. We then use the witness-map perspective of the arrow topos to motivate certified concurrent implementations. An explicit partially open runner hides private state while preserving interleaving with a shared-state environment. The concluding problem is to formulate and, where possible, represent this indexed form of concurrency-aware Sweedler running. Established results are distinguished throughout from new calculations and open proposals.

1 Why another part of the tutorial?

A monad (T, η, μ) describes the substitution of effectful computations into the continuations of other computations. This is a compelling representation of sequential effects; it does not, on its own, tell us how to execute two effects concurrently. Introducing a syntactic command named *fork* is not the same as providing a compositional interpretation of parallel execution. Concurrency adds another mode of composition and, crucially, a comparison between its interaction with sequencing.

This part belongs after the discussion of changed ambient worlds and witnessed systems and before the existing Part VII. It is an independently numbered interlude, so that cross-references throughout the established tutorial need not change. The subject is the algebra of concurrent theories and their stateful implementations, rather than a general account of concurrent geometry or continuous dynamics. It assumes the elementary runner equation

$$\theta_X : TX \times S \longrightarrow R(X \times S), \qquad T \longrightarrow \mathsf{St}_S(R), \qquad \mathsf{St}_S(R)(X) = \big(R(X \times S)\big)^S. \tag{1}$$

Here θ must respect monad unit and multiplication. Whenever the ordinary measuring comonad $\{T, R\}$ exists under the hypotheses established elsewhere in the tutorial, a runner on S is represented by a $\{T, R\}$-coalgebra. We shall discover that these requirements do not suffice for concurrent behaviour.

There are three different pieces of structure throughout this part: (i) the second monoidal structure on an *ambient* duoidal category, used in Sweedler measurement; (ii) a parallel composition on a *particular theory* T; and (iii) evidence or witnesses that an *implementation* satisfies a specification. They must not be conflated.

2 Concurrent monoids: middle-four interchange

Definition 2.1 (Concurrent monoid [1, 3])**.** A concurrent monoid is a poset $(M, \leq)$ with monotone monoids $(M, ;, 1)$ and $(M, \parallel, e)$ satisfying

$$\begin{gathered} 1 \leq e, \qquad e; e \leq e, \qquad 1 \leq 1 \parallel 1, \\ (p \parallel q); (r \parallel s) \leq (p; r) \parallel (q; s). \end{gathered} \tag{2}$$

It is *normal* if $1 = e$.

The last comparison is called *middle-four interchange*. In trace-inclusion models the left side performs a parallel first stage and then a parallel second stage, while the right side permits two threads to progress independently and therefore can admit more schedules. This reading is *not* intrinsic to the order: in the duration model below the reversed numerical order instead compares performance. The normal, common-unit form of concurrent algebra goes back to Hoare–Möller–Struth–Wehrman; here we use the more general two-unit formulation presented by Rivas–Uustalu.

2.1 What do the two units mean?

The sequential unit 1 satisfies $1; p = p = p; 1$: it represents doing nothing *in a sequence*. The parallel unit e satisfies $e \parallel p = p = p \parallel e$: it represents doing nothing *alongside* another process. Although these are both plausible notions of inactivity, the definition does not identify them. In an ordered semantics, the comparison $1 \leq e$ says that the sequentially empty behaviour is below the parallelly empty behaviour. Its precise operational reading depends on what the order measures.

The three unit inequalities are not independent curiosities. In fact, interchange already forces the comparison between the units.

Lemma 2.2 (Interchange compares the units)**.** *For two monotone monoids satisfying middle-four interchange,* $1 \leq e$*, even without assuming the three unit inequalities separately.*

Proof. Using the unit laws for both monoids,

$$1 = 1; 1 = (e \parallel 1); (1 \parallel e) \leq (e; 1) \parallel (1; e) = e \parallel e = e.$$

□

The other two inequalities then imply rather stronger statements.

Proposition 2.3 (Cross-idempotence of the units)**.** *In every concurrent monoid,*

$$\boxed{e; e = e, \qquad 1 \parallel 1 = 1.}$$

Proof. Since $1 \leq e$, monotonicity yields $e = 1; e \leq e; e$, which combines with $e; e \leq e$. Likewise, $1 \parallel 1 \leq e \parallel 1 = 1$, which combines with $1 \leq 1 \parallel 1$. □

Thus each unit is *idempotent* for the other composition, without necessarily being its identity. This is a useful way to understand nonnormality. Indeed, right and left sequential composition with the parallel unit give maps

$$C_R(p) = p; e, \qquad C_L(p) = e; p.$$

They are monotone, extensive ($p = p; 1 \leq p; e$ and $p = 1; p \leq e; p$), and idempotent by $e; e = e$. Hence both are *closure operators* on the underlying poset. They need not

agree. When $1 = e$, they are both the identity. Nonnormality can therefore be read as the possibility that treating parallel inactivity as sequential activity performs a genuine closure or relaxation. Interpretations in terms of silent administrative steps or stuttering are suggestive, but require a particular ordered model; they are not consequences of the axioms alone.

There is also a categorical reason for exactly these three laws. With sequential product written $\otimes =$; and parallel product $\star = \|$, the duoidal unit comparisons have shapes

$$I \longrightarrow J, \qquad J \otimes J \longrightarrow J, \qquad I \longrightarrow I \star I.$$

In a thin category they are precisely $1 \leq e$, $e; e \leq e$ and $1 \leq 1 \parallel 1$. *Normality* means that the first comparison is invertible, i.e. $1 = e$ in a poset; it does not make middle-four interchange invertible. The duration and shuffle examples below are normal but can have strict interchange.

The distinction persists for concurrent monads: the ordinary monad unit preserves pure return, while the parallel unit supplies an empty parallel computation. An ordinary runner preserves the former, but nothing in its monad-morphism axioms guarantees preservation of the latter or of its comparison with pure return. Those are additional obligations for a concurrent runner.

Example 2.4 (A nonnormal three-element concurrent monoid)**.** Take the chain $\bot < s < \top$, with parallel product $x \parallel y = x \wedge y$ and parallel unit $e = \top$. Define sequential composition by

$$x; y = \begin{cases} y, & x = s, \\ x, & x \in \{\bot, \top\}. \end{cases} \qquad 1 = s.$$

Thus $\bot$ and $\top$ are left zeros for sequencing, whereas s is its unit. Both products are associative and monotone. The unit inequalities hold, and a check on the three possible values in each variable verifies middle-four interchange. In this example

x	$\bot$	s	$\top$
$C_R(x) = x; \top$	$\bot$	$\top$	$\top$
$C_L(x) = \top; x$	$\top$	$\top$	$\top$

so $C_R \neq C_L$. This demonstrates that the two closures need not coincide. An interpretation of these elements as abort, skip, and chaos is suggestive, but is not forced by the algebra.

Proposition 2.5 (Normality and invertible interchange)**.** *In a normal concurrent monoid, $p; q \leq p \parallel q$ and $q; p \leq p \parallel q$. If interchange holds with equality everywhere, then normality follows automatically; moreover the two operations coincide and are commutative.*

Proof. In the normal case,

$$p; q = (p \parallel 1); (1 \parallel q) \leq (p; 1) \parallel (1; q) = p \parallel q,$$

and symmetrically $q; p \leq p \parallel q$. If interchange is an equality, then first

$$1 = 1; 1 = (e \parallel 1); (1 \parallel e) = (e; 1) \parallel (1; e) = e \parallel e = e.$$

The two unital operations thus have a common unit and satisfy strict interchange. The Eckmann–Hilton argument makes them equal and commutative. □

Noninvertibility is indispensable if parallel and sequential composition are to remain distinct; commutative effects, for which the two genuinely coincide, are not thereby excluded. In a thin category the comparisons are inequalities; higher settings may retain distinct witnesses of them.

3 A duoidal reminder: ambient interchange is not concurrent execution

Recall a duoidal category $(\mathcal{D}, \otimes, I, \star, J)$ with an interchange of the convention

$$(A \star B) \otimes (C \star D) \longrightarrow (A \otimes C) \star (B \otimes D), \tag{3}$$

and unit maps $I \to I \star I$, $J \otimes J \to J$, and $I \to J$. The ambient interchange provides compatibility of two products; it is not itself a definition of parallel computation for an arbitrary monad T.

The simplest illustration is the canonical map

$$(A \times B) \sqcup (C \times D) \longrightarrow (A \sqcup C) \times (B \sqcup D).$$

It is an injection whose image consists of the two unmixed summands. Hence cardinality gives $ab + cd \leq (a+c)(b+d)$. But the map $1 \sqcup 1 \to 1$ is a non-injective fold, and cardinality does not send arbitrary set maps to inequalities. Thus $(\mathbb{N}, \leq, +, 0, \times, 1)$ does *not* satisfy the required duoidal unit comparison $1 + 1 \leq 1$. This small distinction will matter when we compare rigs, quantitative effects and concurrent operations: middle-four inequality alone is not sufficient.

The normal duration example below uses a different pair of operations and has matching units. In the ordered accessible endofunctor construction of Rivas–Uustalu, concurrent monads become concurrent-monoid objects for composition and ordered Day convolution. The essential added ingredient is a second monoid structure *on* T, with directed compatibility, rather than just the existing ambient convolution product [3, 5].

4 Concurrent monads and their ordered compatibility

Let $(\mathcal{C}, \otimes, I)$ be a suitable ordered monoidal category. A concurrent monad consists of an ordered monad (T, η, μ) and a lax-monoidal structure

$$\psi^0 : I \to TI, \qquad \psi_{X,Y} : TX \otimes TY \longrightarrow T(X \otimes Y),$$

with the required ordered compatibility axioms. In a strictified notation, the two central comparisons are

$$\eta_{X \otimes Y} \leq \psi_{X,Y}(\eta_X \otimes \eta_Y), \tag{4}$$

$$\mu_{X \otimes Y}\, T\psi_{X,Y}\, \psi_{TX,TY} \leq \psi_{X,Y}(\mu_X \otimes \mu_Y). \tag{5}$$

The unit comparisons at I complete these equations; as in (2), their precise orientation is part of the definition, not something that can be inferred from (5) alone.

If $f : X \to TY$ and $g : U \to TV$ are Kleisli arrows, their parallel composite is

$$f \parallel g := \psi_{Y,V}(f \otimes g) : X \otimes U \longrightarrow T(Y \otimes V).$$

Inequality (5) becomes weak middle-four interchange for Kleisli composition. The distinction between *monad multiplication* and *parallel multiplication* is structural, not merely terminological.

5 First example: parallel durations

Paquet–Saville's Example 6.2 takes the locally ordered 2-category **Poset**, the decreasing natural numbers $\mathbb{N}_{\geq} = (\mathbb{N}, \geq)$, and the writer 2-monad $\mathbb{N}_{\geq} \times -$. Sequential grades add; parallel grades combine by maximum. The numerical inequality is

$$\max(n+m, k+\ell) \ \leq_{\mathbb{N}} \ \max(n,k) + \max(m,\ell). \tag{6}$$

Thus two independent threads can finish no later than a design that introduces a barrier between stages, under the idealized duration model. For instance

$$\max(2+5, 4+1) = 7 < 9 = \max(2,4) + \max(5,1).$$

The 2-cell points in the direction specified by the *reversed* hom-order $\mathbb{N}_{\geq}$. In the shuffle model, larger means *more permitted schedules*; here $a \leq_{\mathbb{N}_{\geq}} b$ means $a \geq_{\mathbb{N}} b$, so moving upward in the categorical order means a *shorter completion time*. The direction of a safety or refinement claim must therefore be fixed anew for each semantics.

The underlying normal concurrent monoid uses the two operations $+$ and max, both with unit 0. This resembles max-plus tropical arithmetic but is not the same unital rig: the usual completed tropical rig adjoins $-\infty$ as the additive zero (unit of max), while 0 remains the multiplicative unit (unit of $+$). Here the empty parallel job has duration 0, whereas tropical addition's empty collection of alternatives has value $-\infty$.

Proposition 5.1 (Idempotent semirings as concurrent monoids)**.** *Let $(R, \oplus, 0, \otimes, 1)$ be an idempotent semiring, with natural order $a \sqsubseteq b$ iff $a \oplus b = b$. In the* reversed *order $\sqsupseteq$, put $a; b = a \otimes b$ and $a \parallel b = a \oplus b$, with parallel unit $e = 0$. This is a concurrent monoid. It is normal precisely when $0 = 1$ (in which case the semiring is trivial).*

Proof. The sequential and parallel operations are monotone in the natural order and hence in its reverse. Distributivity and monotonicity of $\oplus$ give

$$(a \oplus b) \otimes (c \oplus d) = ac \oplus ad \oplus bc \oplus bd \sqsupseteq ac \oplus bd,$$

which is middle-four interchange in the reversed order. The unit comparisons become $1 \sqsupseteq 0$, $0 \otimes 0 \sqsupseteq 0$, and $1 \sqsupseteq 1 \oplus 1$; they hold because 0 is least in the natural order, is absorbing, and $\oplus$ is idempotent. Finally $0 = 1$ forces every element $x = x \otimes 1 = x \otimes 0 = 0$. □

For completed max-plus tropical arithmetic, $(\mathbb{N} \cup \{-\infty\}, \max, -\infty, +, 0)$, the two induced closures $C_R(x) = x + (-\infty)$ and $C_L(x) = (-\infty) + x$ are both the constant $-\infty$ map. The duration example is closely related but has 0 as the unit of *both* operations and lacks an absorbing additive zero. Thus max plays two different roles: maximum of simultaneous durations, versus aggregation of alternative paths.

This calculation connects with the tutorial's temporal grading and amortisation constructions. It does *not* by itself produce a concurrent graded Moore monad: one must lift the parallel operation to the relevant graded substitution structure and check directed interchange there.

6 Second example: traces and shuffles

Let A be an alphabet and A^* its free monoid. For finite languages $P, Q \subseteq A^*$, put

$$P; Q = \{uv : u \in P, v \in Q\}, \qquad P \parallel Q = \bigcup_{u \in P, v \in Q} \mathrm{Sh}(u, v),$$

where $\mathrm{Sh}(u, v)$ consists of words preserving the relative order of the letters of each of u and v. With inclusion order and common unit $\{\epsilon\}$, this is a normal concurrent monoid. Here $P \leq Q$ means that Q permits at least all schedules in P; consequently, safety of Q entails safety of P, but not conversely.

Proposition 6.1 (Shuffle interchange)**.** *For all words u, v, r, s,*

$$\mathrm{Sh}(u, v) \cdot \mathrm{Sh}(r, s) \subseteq \mathrm{Sh}(ur, vs). \tag{7}$$

The inclusion can be strict.

Proof. A concatenation of a shuffle of u, v with a shuffle of r, s preserves the internal orders of ur and vs. For $u = a$, $v = b$, $r = c$, $s = d$, the left set has four words while the right has six. In particular, *acbd* and *bdac* occur only on the right. □

The left imposes a barrier after the a, b stage; the right has just the dependencies $a < c$ and $b < d$. A set of words records possible linearizations, not a specified independence witness. This is one motivation for the later study of event structures and precubical sets.

Paquet–Saville obtain a 2-dimensional version using the writer construction $T_A X = X \times A^*$ in **Rel**. The parallel transformation is a *relation* defined by the shuffles, and interchange is an inclusion of relations. Taking Kleisli endomorphisms of the singleton recovers the ordered monoid of languages. This is a particularly economical meeting point of the ordered and bicategorical accounts [4].

7 Sequential interpretations need not preserve concurrency strictly

We can calculate the failure already on finite-language monoids, without importing an additional non-determinism monad. Let $h : A \to B^*$ implement each abstract atomic action by a word of concrete actions, and extend it to the monoid homomorphism $h^* : A^* \to B^*$. Write h_* for direct image on finite languages.

Proposition 7.1 (Block substitution is oplax for shuffle)**.** *The map h_* strictly preserves sequential concatenation and its unit, and satisfies*

$$h_*(P \parallel Q) \;\subseteq\; h_*(P) \parallel h_*(Q). \tag{8}$$

Equality is not required.

Proof. Strict sequential preservation follows from $h^*(uv) = h^*(u)h^*(v)$. A shuffle of abstract letters becomes an interleaving of their concrete implementation *blocks*, hence is among the interleavings of the concrete words when the blocks may also interleave internally. This proves inclusion. For $h(a) = a_1 a_2$ and $h(b) = b_1 b_2$, the image of $a \parallel b$ contains just $a_1 a_2 b_1 b_2$ and $b_1 b_2 a_1 a_2$, while $h(a) \parallel h(b)$ has six interleavings. □

The laxity witnesses the loss of atomicity: implementing a formerly indivisible command by several smaller steps introduces new opportunities for interference. A strict sequential interpretation therefore says less than a concurrent one. The chosen direction (8) describes *more* possible concrete schedules, not necessarily a safety refinement. Safety of a smaller trace set does not imply safety of a larger one.

8 Shared-state resumptions

Fix a discrete state set S. Rivas–Uustalu's finite resumption trees have the recursive description

$$\mathsf{Res}_S X \cong X + (S \to \mathrm{List}(S \times \mathsf{Res}_S X)). \tag{9}$$

The order is induced by the order on X and *order-preserving inclusion of lists*: a list embeds as a subsequence of a larger list, respecting the order of corresponding elements. Multiplicities and the order of alternatives therefore matter. A return is $\mathrm{Ret}(x)$. A one-step node is $\mathrm{Grab}(k)$, with $k(s)$ a finite ordered list of successor states and continuations. Kleisli substitution grafts computations onto returns, retaining the Grab nodes.

The parallel operation merge recursively offers one step from the left operand or one step from the right. For two nonterminal nodes, schematically,

$$\begin{aligned}\mathrm{Grab}(k) \parallel \mathrm{Grab}(\ell) = \mathrm{Grab}(s \mapsto [(s', r \parallel \mathrm{Grab}(\ell)) : (s', r) \in k(s)] \\ +\!+ [(s', \mathrm{Grab}(k) \parallel t) : (s', t) \in \ell(s)]).\end{aligned}$$

The omitted terminal clauses are

$$\begin{aligned}\mathrm{Ret}(x) \parallel \mathrm{Ret}(y) &= \mathrm{Ret}(x, y),\\ \mathrm{Ret}(x) \parallel \mathrm{Grab}(\ell) &= \mathrm{Grab}(s \mapsto [(s', \mathrm{Ret}(x) \parallel r) : (s', r) \in \ell(s)]),\\ \mathrm{Grab}(k) \parallel \mathrm{Ret}(y) &= \mathrm{Grab}(s \mapsto [(s', r \parallel \mathrm{Ret}(y)) : (s', r) \in k(s)]).\end{aligned}$$

The list concatenation matters: ordering and multiplicity of alternatives need not be erased. Each newly scheduled step consults the *current shared state*, including any change made by the other thread. Rivas–Uustalu establish the concurrent structure with its ordered laws [3].

8.1 Running against a particular state

Their residual trees are

$$\mathsf{RRes}_S X \cong X + S \times \mathrm{List}(S \times \mathsf{RRes}_S X).$$

They define

$$\mathrm{run}(\mathrm{Ret}(x), s) = \mathrm{RRet}(x), \tag{10}$$

$$\mathrm{run}(\mathrm{Grab}(k), s) = \mathrm{RGrab}(s, [(s', \mathrm{run}(r, s')) : (s', r) \in k(s)]). \tag{11}$$

The resulting residual monad need not carry the original concurrent operation. At a Grab node, its state-response function has been *evaluated*; an independently recorded residual tree generally cannot be resumed against a different state introduced by a concurrent environment.

For our ordinary Sweedler runner, terminal states should be returned as well:

$$\widehat{\mathrm{run}}(\mathrm{Ret}(x), s) = \mathrm{RRet}(x, s), \tag{12}$$

$$\widehat{\mathrm{run}}(\mathrm{Grab}(k), s) = \mathrm{RGrab}(s, [(s', \widehat{\mathrm{run}}(r, s')) : (s', r) \in k(s)]). \tag{13}$$

Under the conventional finite-tree definitions and state-transformer multiplication, structural induction gives the unit and Kleisli-extension equations for

$$\widehat{\mathrm{run}}_X : \mathsf{Res}_S X \times S \longrightarrow \mathsf{RRes}_S(X \times S).$$

This is an *ordinary* residual stateful runner. The explicit terminal state ensures that a sequential continuation can start from the state reached by the previous computation; it does not restore the ability to react to unseen environmental interference.

9 Why isolated observation is not a concurrent congruence

The exact obstruction can be seen on two states. Let $S = \{0, 1\}$. Define one-step operations put_i that replace any state by i, and get and zero that both preserve the state but return, respectively, the current state and the constant 0. Each is represented by a suitable Grab node followed by a return.

Set

$$p = \mathrm{put}_0; \mathrm{get}, \qquad q = \mathrm{put}_0; \mathrm{zero}, \qquad w = \mathrm{put}_1.$$

The semicolon here denotes sequential Kleisli composition with its evident result types. Importantly, p and q each contain *two schedulable steps.*

Proposition 9.1 (All-state isolated observation is not a parallel congruence)**.** *For every initial state s,* $\mathrm{run}(p, s) = \mathrm{run}(q, s)$ *as residual trees. Nevertheless, the parallel composites $p \parallel w$ and $q \parallel w$ have different isolated residual behaviours, even when both begin in state* 0*.*

Proof. In isolation, the first step always changes the state to 0, so the second step gives the same result in either computation, with the same intermediate and final state labels. Under parallel composition the scheduler can interleave the three steps in the order put_0, put_1, get/zero. The final read in $p \parallel w$ then returns 1, whereas the corresponding step of $q \parallel w$ always returns 0. Thus the parallel residual trees differ. This distinction survives even if one records only the possible terminal results, rather than the full trees. □

The conclusion is stronger than failure of single-state observability. Even the *family* of isolated observations over all starting states omits counterfactual responses of continuations to states reached only through external interference.

To repair this, choose a class $\mathcal{K}$ of well-typed contexts *containing the identity context* $[-]$ and closed under context substitution, and define

$$p \approx_{\mathcal{K}} q \quad \Longleftrightarrow \quad \mathrm{run}(C[p], s) = \mathrm{run}(C[q], s) \quad \text{for all } C[-] \in \mathcal{K} \text{ and } s \in S. \tag{14}$$

This is a congruence for the operations represented by the contexts, and it is the largest such congruence contained in the chosen isolated observational equivalence. The abstract contextual-closure assertion is standard. In particular, the intrinsic characterization problem for shared-variable languages is not new: Brookes's fully abstract transition-trace semantics already addresses it for set-valued observations. The more specific question here is whether a comparably intrinsic, fully abstract characterization survives the *list-sensitive* ordering and multiplicity of Rivas–Uustalu resumptions.

9.1 Brookes's transition traces and their possible refinement

Brookes's fully abstract semantics for a shared-variable parallel language uses *transition traces*, finite sequences of state-change pairs

$$(s_0, t_0)(s_1, t_1) \cdots (s_n, t_n).$$

The gap from t_i to s_{i+1} leaves room for interference from the environment. Closure under *stuttering* and *mumbling* (respectively inserting an idle state pair and combining adjacent compatible pairs) helps make the semantics compositional for the language studied there [2]. Brookes proves full abstraction: trace denotations agree exactly when commands cannot be distinguished in the relevant program contexts. The operational interpretation and closure conditions depend on the language and observation notion; they should not be assumed to transfer unchanged to resumptions.

Rivas–Uustalu already compare their resumption model with a model based on collections of transition traces, and exhibit a concurrent-monad morphism from resumptions to that trace model, preserving parallel composition oplaxly [3]. Their trace construction requires finite multisets for certain concurrent equations: lists retain too much ordering for those particular laws. This pinpoints the question for us: *can one construct a Brookes-style contextual semantics that retains the order and multiplicity of resumption branches, or must a quotient be taken?* A related, separate test is whether stutter/mumble closure can actually arise as one of the nonnormal unit-induced operators $C_R(p) = p; e$ or $C_L(p) = e; p$ in a suitable trace model. No such identification has been established here.

10 Interference-aware semantics

As in the Aczel–Brookes tradition, a useful operational presentation marks environmental changes explicitly. The following is a rely-indexed variation of this familiar idea, not a claim to have invented interference traces. For a rely relation $E \subseteq S \times S$, use configurations (r, s) with two forms of labelled step:

$$\begin{aligned} (r, s) &\xrightarrow{\text{env}} (r, t) && \text{if } sEt, \\ (\mathrm{Grab}(k), s) &\xrightarrow{\text{sys}} (r', s') && \text{if } (s', r') \in k(s). \end{aligned}$$

The first permits an environment step between any two computational steps; the second performs one step of the suspended computation. This supplies an operational model for rely–guarantee interference, but it is not yet a proof of a residual monad or a concurrent measuring object.

For two components with guarantees G_1, G_2 and rely relations E_1, E_2, familiar compatibility conditions are

$$G_1 \subseteq E_2, \qquad G_2 \subseteq E_1.$$

A safety invariant $P \subseteq S$ is preserved by arbitrary finite interleavings if every enabled elementary transition of each component carries states of P back into P. This simple induction is an important baseline: it verifies all schedules without identifying them. Stronger specifications—noninterference, response properties and contextual equivalence—need more than mere invariant closure.

11 Witnesses rather than predicates alone

The original predicate-based safety development of the tutorial was strengthened in its later arrow-topos treatment. Rather than retaining only a mono $P \hookrightarrow S$, work with an arbitrary witness map

$$p : W \longrightarrow S$$

in $\mathbf{Set}^{\mathbf{2}}$. A state can have zero, one or many witnesses. A certified transition is a commuting square

$$\begin{array}{ccc} W & \xrightarrow{\widehat{f}} & W \\ \downarrow p & & \downarrow p \\ S & \xrightarrow{f} & S. \end{array}$$

Its top arrow describes *how evidence is transformed*, not merely whether an assertion remains true. The same principle underlies certified wiring: witnesses of global admissibility and component guarantees construct witnesses of the assumptions required by the components.

In concurrent systems, this is more natural than attaching an independent Boolean predicate to each transition. The parallel multiplication itself may need a lift to witnessed computations, and so may the directed interchange comparison. We must nevertheless distinguish witnesses for *particular executions* from certificates asserting *universal safety*. If $B_L \subseteq B_R$, an execution possible in B_L remains possible in B_R, but safety of all B_L-executions gives no certificate for the extra B_R-executions.

Research problem 11.1 (Witnessed concurrent lifting)**.** *Given an ordered concurrent monad T and a category or fibration of witnesses over its base, determine when its sequential monad structure, parallel multiplication and directed interchange admit coherent witnessed lifts. Develop the corresponding rely–guarantee criteria for combining certified components, and identify a suitable ordered or bicategorical category of these liftings.*

In a locally posetal formulation there is at most one comparison 2-cell between any two chosen 1-cells. A bicategorical formulation may retain distinct proofs of simulation, certification or scheduling refinement. Accordingly, a strict Cat-enriched approach can be a useful first calculation, without prejudging the eventual need for weak bicategorical coherence.

12 Concurrent pseudomonads and true concurrency

Paquet–Saville develop concurrent pseudomonads on monoidal bicategories. Their directed structural modifications make possible a coherent weak interchange without collapsing parallel and sequential composition. They show that a concurrent pseudomonad is strong and that its Kleisli bicategory acquires a lax normal bifunctorial parallel operation. Their concurrent-games example uses event structures with actual causal dependencies, rather than merely sets of interleaving words [4].

The more recent Gunnarsson–Rivas–Uustalu account identifies ordered concurrent monads with monads in a 2-category of ordered monoidal categories, ordered lax-monoidal functors and oplax-monoidal natural transformations. The authors note that the ordinary Kleisli construction need not live in that same 2-category and discuss a larger lax-functorial setting whose coherence is still to be settled [5].

These two developments suggest a controlled progression: ordered comparisons, then 2-dimensional comparison witnesses, then genuinely weak bicategorical composition. They also connect with the tutorial's actegorical and Tambara-module investigations, since strength is related to actions on Kleisli bicategories. No general theorem yet identifies concurrent runners with Tambara modules; that is a separate question.

The event-structure example is mentioned here only to show that the algebraic axioms have genuinely concurrent models. The geometrical treatment is deferred: a subsequent study of precubical sets and directed paths should distinguish higher-dimensional independence witnesses from mere equalities of interleavings. Smooth flows, Lie brackets, trajectory spaces and directed homotopy likewise belong to later parts, where their requisite geometry can be developed properly.

13 Toward concurrent Sweedler measurement

Ordinary Sweedler theory classifies monad-preserving stateful implementations. An ordinary runner has the form $\theta : T \to \mathsf{St}_S(R)$, where the state-transformer closes the chosen state S within each computation. This is exactly why independently run resumptions can lose the counterfactual responses needed for shared-state interference. Merely adding a lax parallel

comparison to $\mathsf{St}_S(R)$ may be the wrong first formulation: there need not be a suitable shared-state parallel operation there.

13.1 A worked partially open runner

The obstruction of §9 suggests that one should not close *all* of the state before composing with an environment. We can make this precise for finite resumptions. Let $S = P \times H$, where P is private and H is shared. Define the partially open computation monad

$$\mathsf{Open}_{P,H} X = \mathsf{St}_P(\mathsf{Res}_H)X = P \longrightarrow \mathsf{Res}_H(X \times P). \tag{15}$$

Unlike isolated execution on $P \times H$, this codomain still has Grab nodes with a live argument $h \in H$: a concurrent environment can change h between two steps. The private state is threaded by the runner, but the *steps* of the component are not eliminated.

Define a family $\Theta_X : \mathsf{Res}_{P\times H} X \to \mathsf{Open}_{P,H} X$ recursively by

$$\Theta(\mathrm{Ret}(x))(p) = \mathrm{Ret}(x, p), \tag{16}$$

$$\Theta(\mathrm{Grab}(k))(p) = \mathrm{Grab}\Big(h \mapsto [(h', \Theta(r')(p')) : ((p', h'), r') \in k(p, h)]\Big). \tag{17}$$

The bracket in (17) preserves the order and multiplicity of the original list; it is not a set comprehension. For every possible shared state h, the next transition of the source is evaluated at (p, h) and its private successor p' is supplied to the translated continuation.

Proposition 13.1 (Partial hiding as an ordinary runner)**.** *For finite well-founded resumption trees, Θ is a monad morphism*

$$\mathsf{Res}_{P\times H} \longrightarrow \mathsf{St}_P(\mathsf{Res}_H).$$

Thus, under the usual representing hypotheses, P carries an ordinary $\{\mathsf{Res}_{P\times H}, \mathsf{Res}_H\}$-coalgebra.

Proof. Naturality is immediate since return values are left unchanged. The unit equation is (16). For the Kleisli extension equation, prove by structural induction on r that

$$\Theta(r \gg\!= f)(p) = \Theta(r)(p) \gg\!= \big((x, p') \mapsto \Theta(f(x))(p')\big).$$

At a return this is the unit equation; at a Grab both sides have the same list of H-successors and the induction hypothesis identifies the corresponding subtrees. This is precisely the multiplication law for the state transformer on the right. □

The relevant concurrency claim is deliberately *asymmetric*. Let $j : \mathsf{Res}_H Y \to \mathsf{Res}_{P\times H} Y$ lift a computation which never reads or changes the private state:

$$\begin{aligned} j(\mathrm{Ret}(y)) &= \mathrm{Ret}(y), \\ j(\mathrm{Grab}(\ell)) &= \mathrm{Grab}\big((p, h) \mapsto [((p, h'), j(q')) : (h', q') \in \ell(h)]\big). \end{aligned}$$

For $u \in \mathsf{Res}_H(X \times P)$ and $q \in \mathsf{Res}_H Y$, put

$$u \mathbin{\widehat{\|}} q := \mathsf{Res}_H(\alpha)(u \parallel_H q), \qquad \alpha : ((X \times P) \times Y) \cong (X \times Y) \times P.$$

The right-hand side again has a *single* shared-state response function at every step.

Proposition 13.2 (External concurrency survives hiding)**.** *For $r \in \mathsf{Res}_{P\times H}X$, $q \in \mathsf{Res}_H Y$, and $p \in P$,*

$$\Theta(r \parallel_{P\times H} j(q))(p) = \Theta(r)(p) \mathbin{\widehat{\parallel}} q, \tag{18}$$

up to the indicated canonical reordering of return values. The equality holds for the full ordered-list trees, not just their sets of terminal outcomes.

Proof. Induct on the pair of finite trees. If a step of r is scheduled first, its source transition at (p, h) produces (p', h') and r'; both sides emit an H-step to h' and continue with $\Theta(r')(p')$ and q. If a step of $j(q)$ is scheduled first, it changes h but leaves p fixed; both sides leave $\Theta(r)(p)$ ready to respond to the new shared state. The left-first and right-first alternatives are concatenated in the same order. The return cases follow from the corresponding clauses of $\parallel$. □

Example 13.3 (A private update racing with a shared write)**.** Let $P = H = \{0, 1\}$, initially $(p, h) = (0, 0)$. The component r first sets $p := 1$ in one Grab step and then reads h in another step. The environment q sets $h := 1$ in one step. All interleavings respect the local order of r, so there are three schedules:

schedule	returned read	final (p, h)
private write; read; shared write	0	$(1, 1)$
private write; shared write; read	1	$(1, 1)$
shared write; private write; read	1	$(1, 1)$.

The partially open runner preserves all three, including the interference between the component's two steps. By contrast, an isolated execution of r from $(0, 0)$ fixes the read at 0, so independently interleaving its completed residual history with q cannot recover the other two outcomes. Notice that *hiding the private value does not erase the private update's* Grab *node*: it remains a schedulable step with an unchanged H-state. Erasing such a step would be another abstraction, potentially governed by stuttering closure in the sense of Brookes (§9.1).

13.2 Where symmetric parallel composition fails

Equation (18) does *not* give a canonical concurrent multiplication on $\mathsf{St}_P(\mathsf{Res}_H)$. If both processes read or change P, evaluating them separately at the same initial private state loses the alternative private-state responses needed after interference. For instance, from $p = 0$, parallelizing a private $\mathrm{put}_P(1)$ with a private get_P permits the read to return 1. But running the get_P independently at $p = 0$ fixes its residual return at 0. The inherited $\parallel_H$ on the two resulting trees cannot restore the missing possibility. This is a failure of the *naive pointwise parallel operation*, not a proof that no concurrent structure on the state transformer could ever be constructed.

The positive construction is an external action

$$\mathsf{Open}_{P,H}X \times \mathsf{Res}_H Y \longrightarrow \mathsf{Open}_{P,H}(X \times Y), \qquad (u, q) \longmapsto (p \mapsto u(p) \mathbin{\widehat{\parallel}} q). \tag{19}$$

Its associativity and unit structure derives from concurrent merge; compatibility with Kleisli composition is *lax*, and an exact actegorical formulation must record those comparisons. This supplies a concrete motivation for the actegory/Tambara questions in Appendix D, without identifying these structures prematurely.

More symmetric composition is possible when private states are *disjoint*. If two components own P_1 and P_2 respectively, but interact with the same shared H, there is an external tensor

$$\begin{aligned} &\mathsf{Open}_{P_1,H}X \times \mathsf{Open}_{P_2,H}Y \longrightarrow \mathsf{Open}_{P_1\times P_2,H}(X \times Y), \\ &(u, v) \longmapsto ((p_1, p_2) \mapsto \mathsf{Res}_H(\beta)(u(p_1) \parallel_H v(p_2))), \end{aligned} \tag{20}$$

where β reorders the two results and private states. Unlike pointwise parallel on a fixed P, this operation is compatible with the source interleaving because each component updates only its own private factor. Its natural setting is therefore *indexed* over state decompositions rather than a single endomorphism category.

There is also a staged-hiding identity. For $S = P \times Q \times H$, recursive hiding of P and then Q agrees, up to canonical reassociation, with hiding $P \times Q$ at once:

$$\mathsf{St}_P(\mathsf{St}_Q(\mathsf{Res}_H)) \cong \mathsf{St}_{P\times Q}(\mathsf{Res}_H).$$

This is the usual state-transformer isomorphism, and the equality of the corresponding runner translations follows by induction on the resumption trees. What changes under further hiding is the class of environments with which external concurrency is preserved: after hiding both P and Q, only H remains available for interference.

For certified systems, the decomposition itself may be witnessed by an arrow $W \to P \times H$, together with guarantees that environmental steps touch only H and preserve the relevant assumptions. A lift of (18) would then transport witnesses for shared-state interactions. Such a witnessed lift is a further proof obligation, not supplied automatically by Proposition 13.2.

Research problem 13.4 (Indexed open concurrent runners)**.** *Determine the indexed or double-categorical universal property of partial hiding, external environmental action* (19)*, and the tensor* (20)*. Characterize compatible witness liftings and their behaviour under staged hiding. Ask only then whether an enriched measuring object represents these structures.*

13.3 A subsequent representation problem

Once the category or bicategory of such open, possibly witnessed runners has been identified, ask whether it is represented by a concurrency-aware measuring object. A candidate symbol is $\{T, R\}_{\mathrm{conc}}$, but neither its existence nor a canonical comparison with $\{T, R\}$ follows from ordinary duoidality.

Research problem 13.5 (Concurrent Sweedler representation)**.** *Specify the residual theory and coherent comparison cells appropriate to open concurrent runners, and determine whether their moduli are represented by a measuring object or comonad. Compare the resulting universal property with ordinary $\{T, R\}$, stating any forgetful comparison only after its variance and ambient category have been established.*

13.4 Scope and handoff to later parts

The results in this interlude concern the *algebra* of parallel and sequential composition and the effect of stateful implementation on that algebra. They are applicable to controllers and distributed physical systems as well as to programs. A controller may schedule independent actuators, impose synchronization barriers, or interact with a shared plant state; these are already instances of the operational distinctions expressed by concurrent

monoids and resumptions. We have deliberately not identified the abstract parallel product with simultaneous physical evolution, nor assumed that scheduling independence entails dynamical independence.

Two later lines of development should remain distinct. The first uses precubical sets, event structures and directed algebraic topology to ask when different execution orders are connected by *specified* independence data, and what obstacles do to directed path spaces. The second uses smooth control systems to ask when physical realizations respect such independence—for instance, through commutation of local flows. Neither geometry is needed to prove the algebraic runner results above. Conversely, neither is supplied automatically by a concurrent monad. They will be tests for an eventual higher, witnessed Sweedler theory, not assumptions in the present construction.

This leaves a concrete agenda for the next stage of *this* subject: refine the partial-hiding construction to list-sensitive interference traces; work out coherent witness transport for the external action; and formulate a universal property for indexed open runners. Those questions can be posed and pursued before enlarging the ambient geometry.

What we can already conclude

1. *Directed interchange is real algebraic information.* It distinguishes scheduling architectures and avoids the Eckmann–Hilton collapse.

2. *Sequential preservation is weaker than concurrent preservation.* Block substitution of atomic actions provides a strict sequential homomorphism with only lax parallel compatibility.

3. *Ordinary running can forget counterfactual state responses.* Isolated residual observations, even from every initial state, are not generally congruent under shared-state parallel composition.

4. *Witnessing belongs inside the implementation problem.* A certificate for a restrictive scheduling architecture need not cover new interleavings introduced by its implementation.

5. *Brookes provides the intrinsic trace benchmark.* The open problem is to retain the list-sensitive information of resumptions under contextual composition.

6. *Partial state hiding has a concrete positive construction.* The runner $\Theta : \mathsf{Res}_{P\times H} \to \mathsf{St}_P(\mathsf{Res}_H)$ preserves interleaving with an H-only environment, although naive symmetric parallel composition of two privately stateful computations fails.

7. *Representation is genuinely open.* Neither ambient duoidality nor the existence of the ordinary Sweedler hom proves that a concurrent measuring comonad exists.

The immediate conclusion is that ordinary Sweedler runners already support a useful asymmetric form of concurrent implementation: private state may be hidden while interaction with a shared-state environment stays open. The next mathematical task is to organize this indexed construction and its witness transport, and only then to test a representing property. The broader control-theoretic and geometric examples are deliberately reserved for later parts.

13.5 Beyond binary concurrency: dependence and logical duality

Normal concurrent monoids with commutative parallel composition are closely related, by reversal of order, to the thin physical duoidal categories of Shapiro and Spivak ([6]). The distinction in orientation is significant: concurrent refinement may order computations by increasing permitted behaviour, whereas physical duoidal categories order dependency structures by increasing causal constraints.

Shapiro and Spivak further distinguish binary series-parallel composition from operations indexed by arbitrary finite dependency posets. Their dependence categories provide such operations, but these are additional structure: different dependence-category extensions of the same physical duoidal category need not agree. In particular, directly defined critical-path semantics may differ from the connected-limit extension derived from binary compositions.

This suggests studying dependence-aware runners that preserve not only sequential and parallel operations, but explicitly chosen dependency-indexed compositions. The process decorations of Shapiro and Spivak provide a possible starting point for such a construction.

A separate logical connection arises from BV and MAV, where noncommutative sequential composition interacts with two multiplicative connectives exchanged by linear negation. This duality relates the opposite orientations of directed interchange. Whether the associated deep-inference normalization theory supplies a useful proof calculus for witnessed runners is left to the investigation of internal languages.

References

[1] C. A. R. Hoare, B. Möller, G. Struth and I. Wehrman, *Concurrent Kleene Algebra and its Foundations*, Journal of Logic and Algebraic Programming 80(6) (2011), 266–296. doi:10.1016/j.jlap.2011.04.005.

[2] S. Brookes, *Full Abstraction for a Shared-Variable Parallel Language*, Information and Computation 127(2) (1996), 145–163. doi:10.1006/inco.1996.0056.

[3] E. Rivas and T. Uustalu, *Concurrent Monads for Shared State*, PPDP (2024), article 17. DOI: 10.1145/3678232.3678249.

[4] H. Paquet and P. Saville, *Effectful Semantics in Bicategories: Strong, Commutative, and Concurrent Pseudomonads*, LICS (2024). `https://arxiv.org/abs/2304.11014`.

[5] B. Gunnarsson, E. Rivas and T. Uustalu, *Concurrent Monads as Monads in a 2-Category*, MSFP (2026). `https://msfp-workshop.github.io/msfp2026/submissions/msfp-final15.pdf`.

[6] B. Shapiro, D. Spivak, *Duoidal Structures for Compositional Dependence*. `https://arxiv.org/abs/2210.01962`