

Getting to Grips with Sweedler Runners for Systems

A conversational tutorial, Part III: interpreting, combining, and varying theories
Revised to include the tensor product of theories

September 2026

Aim. Parts I and II concentrated on the loose direction: a theory is realised through another theory by a stateful runner. Part III adds the tight direction. A tight map directly interprets one theory in another, without introducing runner state. Wiring is one important instance, but tight maps also rename, forget, compile, quotient, compare, and reparameterise theories. Once this is combined with the limits, colimits, and tensor product available for Lawvere theories, and with families such as $T_{\text{Moore}}(A, O)$, a much larger structural calculus appears.

Contents

1	The second direction in Run	2
2	Tight maps are not just wiring diagrams	3
2.1	One tick becomes several ticks	3
2.2	A primitive becomes a composite term	4
2.3	Forgetting richer structure	4
2.4	Quotients and adding equations	4
3	Where wiring enters: coproducts of theories	5
3.1	On handlers: the component operations must already share a carrier	5
3.2	On comodels: product state spaces are much more natural	5
3.3	A tiny serial example	6
3.4	Concrete wiring: closing a Moore machine with a Mealy machine	7
4	A categorical calculus of theories	10
4.1	Coproduct: put capabilities side by side	10
4.2	Coequalizer: impose equations categorically	10
4.3	Pushout: glue theories along a shared subtheory	11
4.4	Equalizer: where two interpretations agree	11
4.5	Products and pullbacks: compatible target interpretations	12
4.6	Tensor product: make two theories commute	12
5	Why completeness and cocompleteness matter	15
6	And what about Poly?	15
6.1	A caution about polynomial monads	16

7 Theories that vary with parameters	16
7.1 Input maps are covariant	16
7.2 Output maps are contravariant	17
7.3 What parameter change does to an actual Moore machine	17
8 Combining parameterised theories	19
8.1 A span of interfaces gives a span of theories	19
8.2 Parameter arithmetic and theory arithmetic are different	20
8.3 Tensoring parameterised families	20
8.4 Kan extension: transporting whole families of theories	20
9 The total category of Moore-interface theories	21
10 Tight maps and loose runners in one picture	22
10.1 Theory change can happen on either side of a runner	23
10.2 Wire first or realise first?	23
11 Toward an indexed or fibred Run	24
12 A worked parameter-change example	24
13 What Part III has added	25
14 Where to go next	27
References and orientation	27

1 The second direction in Run

Parts I and II used loose arrows, which we wrote

$$T \rightsquigarrow R.$$

A loose arrow carries a state object S and, in the Set-based case, may be represented by a residual runner

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

Its action on handlers changes the underlying carrier:

$$C \longmapsto C^S.$$

The tight direction is much simpler to write. It is an ordinary map of theories or monads

$$f : T \longrightarrow R.$$

No private state is introduced. Instead every operation of T is interpreted directly as a term or operation of R , compatibly with substitution and equations.

If (C, r) is an R -handler, restriction along f gives a T -handler

$$f^*(C, r),$$

on the *same* underlying carrier C . In monad notation its structure map is simply

$$TC \xrightarrow{f_C} RC \xrightarrow{r} C.$$

For Lawvere theories, the same restriction idea applies to comodels: an R -comodel can be regarded as a T -comodel by interpreting the T -cooperations through f . Again, the underlying state set is unchanged.

The contrast to keep in mind is

tight $f : T \rightarrow R$	same carrier; reinterpret the operations
loose $Y : T \rightsquigarrow R \text{ on } S$	stateful realisation; handlers move $C \mapsto C^S$.

A tight map says that T is already expressible in the language of R . A runner says that T can be *realised* using R , with additional state mediating the realisation.

2 Tight maps are not just wiring diagrams

Before using tight maps for wiring, it is worth seeing several examples where the word “wiring” would be misleading.

2.1 One tick becomes several ticks

Let T_{un} be the free theory on one unary operation u . Its monad is

$$\mathbb{N}y.$$

For any $k \in \mathbb{N}$, there is a tight endomorphism

$$q_k : T_{\text{un}} \longrightarrow T_{\text{un}}$$

sending the generator to its k -fold iterate:

$$u \longmapsto u^k.$$

If a T_{un} -handler is an endofunction

$$g : B \rightarrow B,$$

then restriction along q_k gives the handler

$$g^k : B \rightarrow B.$$

Nothing has been wired to another component. We have simply *compiled one primitive step into a macro consisting of k target steps*.

On a comodel $f : S \rightarrow S$, the same interpretation replaces one abstract tick by

$$f^k : S \rightarrow S.$$

This is probably the smallest useful example of a tight map as compilation.

2.2 A primitive becomes a composite term

Let R be freely generated by two unary operations p, q , and let T be freely generated by one unary operation u . A theory map

$$T \longrightarrow R$$

may send

$$u \longmapsto q \circ p \circ p.$$

An R -handler (B, p_B, q_B) is thereby viewed as a T -handler with operation

$$q_B \circ p_B \circ p_B.$$

Again this is better described as a *term interpretation*, *macro expansion*, or *lower-level implementation* than as wiring.

2.3 Forgetting richer structure

A map from a smaller theory into a richer theory lets us forget all of the richer operations not mentioned by the map. For example, choose one input $a_0 \in A$. There is a tight map

$$T_{\text{un}} \longrightarrow T_{\text{Moore}}(A, O)$$

sending

$$u \longmapsto \text{step}_{a_0}.$$

Any Moore handler can therefore be regarded simply as an endofunction by remembering only the operation corresponding to a_0 .

The tight map points from the *smaller interface* to the richer one; the induced functor on handlers goes back the other way.

2.4 Quotients and adding equations

We already met another important tight map in Part I:

$$F(y^O) \twoheadrightarrow y^O,$$

where the target is the reader theory obtained by imposing the reader equations.

A model of the quotient is, by restriction, a model of the free theory. The map itself records the fact that certain previously distinct terms have become equal. Thus tight maps include not only translations of operation symbols but also *passage to theories with stronger equations*.

This example will become even more useful when we discuss coequalizers below.

A tight map can perform at least four recognisably different jobs:

rename/reindex	compile	forget	impose laws.
----------------	---------	--------	--------------

Wiring is another job, but it should not swallow the others conceptually.

3 Where wiring enters: coproducts of theories

Suppose we have theories

$$T_1, \dots, T_n.$$

Their coproduct

$$T_1 + \dots + T_n$$

contains the operations of all the component theories, subject only to the equations internal to each component. No new equations are imposed between operations coming from different summands.

A tight map

$$W : T \longrightarrow T_1 + \dots + T_n$$

can therefore interpret each operation expected by the external theory T as a term built from operations supplied by the component theories. This is the basic reason such maps can be read as wiring diagrams.

But it helps to distinguish the handler and comodel sides.

3.1 On handlers: the component operations must already share a carrier

An algebra for $T_1 + T_2$ is a set carrying both a T_1 -algebra structure and a T_2 -algebra structure. Thus

$$\mathbf{Alg}(T_1 + T_2)$$

may be viewed as the category of pairs of algebra structures on one common carrier, with no interaction equations beyond those already present in the two theories.

Consequently, a map

$$W : T \rightarrow T_1 + T_2$$

restricts such a jointly structured carrier to a T -handler.

There is an important caveat: given an arbitrary T_1 -handler on C_1 and an arbitrary T_2 -handler on C_2 , the theory coproduct by itself does *not* magically produce a $(T_1 + T_2)$ -handler on $C_1 \times C_2$. For general algebraic operations there is no canonical way to let the first theory act on one coordinate while leaving the other untouched.

So on the handler side, “map into a coproduct” is only one ingredient of a general wiring formalism.

3.2 On comodels: product state spaces are much more natural

For stateful machines represented as comodels, the situation is better behaved.

Suppose S_1 is a T_1 -comodel and S_2 a T_2 -comodel. A cooperation of arity I in T_1 has the form

$$S_1 \longrightarrow I \times S_1.$$

It extends canonically to $S_1 \times S_2$ by acting on the first coordinate and carrying the second coordinate through unchanged. Likewise the T_2 -cooperations act on the second coordinate and preserve the first.

Thus $S_1 \times S_2$ canonically carries a $(T_1 + T_2)$ -comodel structure. Restricting this structure along

$$W : T \rightarrow T_1 + T_2$$

produces a T -comodel on the product state space.

This is a very concrete wiring recipe:

combine component state spaces by product, then restrict along W .

There is in fact more structure in this product-state construction. Since the T_1 -cooperations act only on the S_1 -coordinate and the T_2 -cooperations only on the S_2 -coordinate, the two families of cooperations commute. Thus the resulting comodel factors through the tensor product theory

$$T_1 \otimes T_2,$$

not merely through the coproduct $T_1 + T_2$. We return to the systems meaning of this commutation in Section 4.2.

3.3 A tiny serial example

Let T_u, T_v, T_w each be the free unary theory on one generator, and form

$$T_u + T_v.$$

Define

$$W : T_w \longrightarrow T_u + T_v$$

by

$$w \longmapsto v \circ u.$$

Let

$$f : S \rightarrow S$$

be a T_u -comodel and

$$g : R \rightarrow R$$

a T_v -comodel. On $S \times R$, the u -cooperation updates only S , and the v -cooperation updates only R . Therefore the restricted T_w -comodel has one-step transition

$$(s, r) \longmapsto (f(s), g(r)).$$

The tight map W has turned one external operation into a little serial program using two component operations. This is recognisably wiring, but it is wiring understood as *term interpretation*.

For the systems programme, this distinction is useful. Tight maps tell us how the *operations* of the external interface are assembled from component operations. Separate categorical structure is responsible for assembling the *state carriers* or handlers on which those operations live. Comodels make the product-state construction particularly transparent.

Note: there is a reading-order reversal which is easy to miss. Terms $v(u(x))$ or $v \circ u$ is evaluated on an algebra in the usual inside-out order: first u , then v . On a comodel, however, the outer operation v is encountered first, since cooperations unfold the syntax from the root outward. Thus the same term is read in opposite temporal orders on the model and comodel sides.

3.4 Concrete wiring: closing a Moore machine with a Mealy machine

The preceding discussion described wiring abstractly by means of a tight theory map

$$W : T \longrightarrow T_1 + \cdots + T_n.$$

It is worth seeing this mechanism in a familiar systems example. Consider an (A, B) -Moore machine

$$(S, \delta, o),$$

with transition and output maps

$$\delta : A \times S \longrightarrow S, \quad o : S \longrightarrow B,$$

together with a (B, A) -Mealy machine

$$(R, m),$$

where

$$m : B \times R \longrightarrow A \times R.$$

Write

$$m(b, r) = (\lambda(b, r), \rho(b, r)).$$

Thus the Moore machine receives inputs in A and emits outputs in B , while the Mealy machine receives inputs in B and emits outputs in A . The two interfaces therefore fit together in a feedback loop. Closing the loop gives an autonomous dynamical system on the product state space $S \times R$. Starting from (s, r) , the Moore machine first exposes the value

$$b = o(s).$$

The Mealy machine reacts to this value, producing

$$(a, r') = m(b, r),$$

and the resulting a is fed back into the Moore machine:

$$s' = \delta(a, s).$$

One complete closed-system step is therefore

$$(s, r) \longmapsto (\delta(\lambda(o(s), r), s), \rho(o(s), r)).$$

We would like to see this familiar wiring operation arise from the theory calculus of this part.

The Moore theory

Recall that the theory $T_{\text{Moore}}(A, B)$ is generated by unary operations

$$\text{step}_a \quad (a \in A)$$

together with a B -ary operation

$$\text{get},$$

subject to the reader equations on repeated observations. Its comodels are precisely Moore machines:

$$S, \quad \delta : A \times S \rightarrow S, \quad o : S \rightarrow B.$$

In a comodel, the cooperation corresponding to **get** reads the current B -output while leaving the state unchanged, whereas **step** _{a} updates the state according to $\delta(a, -)$.

A theory for the Mealy component

A convenient algebraic presentation of the (B, A) -Mealy interface is to take, for each $b \in B$, an A -ary operation

$$\text{mealy}_b.$$

A comodel then supplies, for each b , a cooperation

$$R \longrightarrow A \times R.$$

Equivalently, these assemble into

$$m : B \times R \longrightarrow A \times R,$$

which is exactly the transition/output map of a Mealy machine. Let us denote this theory by

$$T_{\text{Mealy}}(B, A).$$

The external theory of a closed tick

Now let $\mathbb{N}y$ denote the free theory on one unary operation

$$u.$$

As we saw in Part I, a comodel of $\mathbb{N}y$ is simply a set equipped with an endofunction. Thus $\mathbb{N}y$ is a natural theory for an autonomous discrete-time system: u means “advance the closed system by one tick”. Consider the coproduct theory

$$T_{\text{Moore}}(A, B) + T_{\text{Mealy}}(B, A).$$

There is a tight interpretation

$$W : \mathbb{N}y \longrightarrow T_{\text{Moore}}(A, B) + T_{\text{Mealy}}(B, A)$$

defined by sending the unary generator u to the term

$$W(u)(x) = \text{get}\left(b \mapsto \text{mealy}_b(a \mapsto \text{step}_a(x))\right).$$

This formula is worth reading operationally.

1. The outer **get** asks the Moore component for its current output $b \in B$.
2. The operation mealy_b feeds that value into the Mealy component. Its response selects an $a \in A$ and updates the Mealy state.
3. The selected a determines which unary operation step_a is performed on the Moore component.
4. The result is one complete tick of the closed system.

The wiring diagram has therefore become an ordinary term of the coproduct theory.

Restricting the product comodel

Let S carry the Moore comodel and R the Mealy comodel. The coproduct theory acts naturally on the product state space

$$S \times R :$$

Moore cooperations act on the S -coordinate while leaving R fixed, and Mealy cooperations act on R while leaving S fixed. Restricting this

$$T_{\text{Moore}}(A, B) + T_{\text{Mealy}}(B, A)$$

comodel along the tight map W produces a

$$\mathbb{N}y$$

comodel on $S \times R$. Since a $\mathbb{N}y$ -comodel is just an endofunction, it remains only to calculate that endofunction. Starting from (s, r) , the term $W(u)$ first executes **get**, producing

$$b = o(s).$$

The Mealy cooperation then produces

$$(a, r') = m(b, r),$$

and finally the Moore cooperation step_a sends

$$s \mapsto \delta(a, s).$$

Thus the resulting endofunction is

$$(s, r) \mapsto (\delta(\lambda(o(s), r), s), \rho(o(s), r)),$$

which is exactly the closed Moore–Mealy system written down at the beginning. This gives a concrete instance of the general principle

wiring is a tight interpretation of an external theory inside a composite theory of components.

The external theory here is merely the theory of a single autonomous tick. The internal coproduct theory remembers the two component interfaces, while the tight map W specifies how one external tick is implemented by their interaction.

Why this feedback does not yet require a trace

There is a small but important point hidden in this example. The feedback loop is causal because the Moore output

$$o : S \longrightarrow B$$

depends only on the current Moore state. The current B -output can therefore be supplied to the Mealy machine before the new A -input has been calculated. If both components were genuinely Mealy, with each current output depending instantaneously on the other's current output, closing the ports could produce an algebraic loop. One would then need some additional structure for solving or interpreting the loop: a delay, a chosen fixpoint, a trace, or some other feedback principle. It is therefore useful to distinguish

causal Moore–Mealy feedback : tight wiring of the kind considered here,

instantaneous cyclic feedback : a genuinely traced or wheeled construction.

We shall return to the second phenomenon much later, when discussing traces, wheels and closed executions.

From wiring to regulation. We shall return to exactly this Moore–Mealy feedback loop in Part V, after introducing predicates and certified systems. There, instead of asking only for the closed update

$$h : S \times R \longrightarrow S \times R,$$

we shall equip the product state space with a predicate describing which joint states are acceptable, and ask whether that predicate is preserved by h . This connects directly with a recent “good regulator theorem” of Virgo, Biehl, Baltieri and Capucci [?]. Their setup consists precisely of a Moore agent coupled to a Mealy environment. A regulation task is specified by a good subset of the joint state space, and successful regulation is witnessed by a nonempty forward-invariant subset lying inside it. Thus the same closed-loop wiring constructed above will provide a particularly concrete example of what changes when we pass from sets to 2-valued predicates.

4 A categorical calculus of theories

The category of ordinary finitary Lawvere theories is complete and cocomplete; more generally, the corresponding categories of ranked/enriched Lawvere theories are locally presentable under standard hypotheses. So we are not limited to hand-written presentations. We can build new theories by universal constructions.

This is one of the places where Part III goes decisively beyond wiring diagrams.

4.1 Coproduct: put capabilities side by side

We have already used

$$T + R.$$

It freely combines the two theories without adding mixed equations.

On models, a $(T + R)$ -algebra is a common carrier equipped with both a T -structure and an R -structure. In this precise sense, coproduct means

make both repertoires of operations available.

For comodels, separate component state spaces can be combined by product and allowed to respond independently to operations from the two summands.

4.2 Coequalizer: impose equations categorically

Suppose

$$P \begin{matrix} \xrightarrow{f} \\ \rightrightarrows \\ \xrightarrow{g} \end{matrix} T \xrightarrow{q} Q$$

is a coequalizer in **Law**. Then Q is obtained from T by universally forcing

$$qf = qg.$$

A Q -algebra is equivalently a T -algebra in which the two induced P -structures agree. So coequalizers provide the categorical form of “add these equations.”

The reader theory is the running example. We began with a freely generated branching theory $F(y^O)$ and imposed equations saying, in effect, that a discarded read is inert and repeated reads are coherent. The quotient map

$$F(y^O) \twoheadrightarrow y^O$$

can be organised as a suitable coequalizer of the two sides of those equations.

This is conceptually useful for runners too: Part II showed that equations imposed in the source theory become correctness constraints on admissible implementations. Coequalizers tell us that this passage from “raw syntax” to “lawful interface” is itself a universal construction in the category of theories.

4.3 Pushout: glue theories along a shared subtheory

Now take a span

$$K \xrightarrow{i_1} T_1, \quad K \xrightarrow{i_2} T_2.$$

Its pushout

$$\begin{array}{ccc} K & \xrightarrow{i_1} & T_1 \\ i_2 \downarrow & & \downarrow \\ T_2 & \longrightarrow & T_1 \amalg_K T_2 \end{array}$$

combines T_1 and T_2 while identifying their two copies of K .

A model of the pushout is a T_1 -model and a T_2 -model on one carrier whose induced K -models agree. Thus the pushout means

combine two capabilities while sharing some specified structure.

This is richer than a coproduct. If two component theories each contain a clock subtheory, for example, the coproduct gives two unrelated clocks; the pushout over the common clock theory forces them to be the *same* clock. Likewise one can glue theories along a common store interface, a common observation operation, a common family of ports, or any other shared algebraic fragment.

This begins to look less like a conventional wiring diagram and more like *architectural assembly of theories*.

4.4 Equalizer: where two interpretations agree

Suppose

$$f, g : T \rightrightarrows R.$$

Their equalizer

$$E \xrightarrow{e} T$$

is universal among theory maps into T on which the two interpretations agree:

$$fe = ge.$$

One can therefore read E as the largest universal fragment of T on which two competing semantics, compilations, or implementations into R coincide.

This is potentially useful for comparing implementations: rather than asking whether two interpretations of all of T are equal, one can isolate the subtheory on which they agree.

4.5 Products and pullbacks: compatible target interpretations

Products and pullbacks are easy to misuse if one argues by superficial duality with coproducts and pushouts, so it is better to stay close to their universal properties.

For the product

$$T_1 \times T_2,$$

a tight map

$$Q \longrightarrow T_1 \times T_2$$

is exactly a pair of tight maps

$$Q \rightarrow T_1, \quad Q \rightarrow T_2.$$

Thus the product is a universal *joint target* for two interpretations of the same source theory.

Likewise, given

$$T_1 \rightarrow K \leftarrow T_2,$$

a map

$$Q \longrightarrow T_1 \times_K T_2$$

is a pair of interpretations

$$Q \rightarrow T_1, \quad Q \rightarrow T_2$$

whose composites into K agree.

So a pullback can encode a pair of target descriptions that are required to present the same view at some common abstraction level K .

Colimits of theories often have a very direct “combine generators and impose identifications” reading on models. Limits are often more naturally read through their mapping property: they organise *compatible families of interpretations into several target theories*. We should resist forcing a simpler slogan where the universal property is already the clean explanation.

4.6 Tensor product: make two theories commute

There is another basic way to combine Lawvere theories which is not a categorical product or coproduct. The *Kronecker tensor product*

$$T \otimes R$$

is universal among theories receiving maps

$$T \longrightarrow T \otimes R, \quad R \longrightarrow T \otimes R$$

for which every operation coming from T commutes with every operation coming from R .

The tensor product has a particularly concrete systems interpretation on comodels. Recall the product-state construction of Section 3.2. If S_1 is a T_1 -comodel and S_2 a T_2 -comodel, then

$$S_1 \times S_2$$

carries the evident $T_1 + T_2$ -comodel structure: T_1 -cooperations act on the first coordinate and T_2 -cooperations on the second. These two families of cooperations automatically commute. Consequently the structure factors through the quotient

$$T_1 + T_2 \longrightarrow T_1 \otimes T_2.$$

Thus the canonical product-state machine is not merely a comodel of the coproduct theory; it is canonically a comodel of the tensor product theory.

This gives the tensor a sharper systems meaning. A $T_1 \otimes T_2$ -comodel is a system carrying two interfaces whose operations commute across the two ports. For Moore theories this becomes a non-interference condition.

Suppose, for example, that the first port has input actions

$$\delta_1(a_1, -) : S \rightarrow S$$

and the second port has observation

$$o_2 : S \rightarrow O_2.$$

Commutation of step_{a_1} with get_2 says

$$o_2(\delta_1(a_1, s)) = o_2(s).$$

Thus an input arriving at port 1 cannot change what is observed at port 2. Symmetrically,

$$o_1(\delta_2(a_2, s)) = o_1(s).$$

The other tensor equations express the corresponding commutation of the two transition structures and of the two observation structures.

This is a categorical form of non-interference: actions presented through one interface cannot be detected through the other interface. In the language of Goguen–Meseguer, the tensor equations enforce the algebraic core of the requirement that one class of actions should not influence another class of observations.

The product-state construction

$$S_1 \times S_2$$

is only the simplest example. A comodel of $T_1 \otimes T_2$ need not decompose as a product of two state spaces. The two interfaces may share hidden state, provided their visible actions still satisfy the commutation equations.

For example, suppose the state contains an unobserved counter

$$n \in \mathbb{N}$$

which may be incremented by actions from either port. If both ports act by commuting updates on n , while the observation exposed at each port is invariant under actions from the other, then the resulting machine may still be a $T_1 \otimes T_2$ -comodel even though its state is genuinely shared.

A comodel of $T_1 \otimes T_2$ is not necessarily a product of two independent machines. It is a two-interface machine satisfying cross-interface commutation equations. For Moore interfaces, these equations say in particular that actions at one port cannot affect observations at the other.

It is helpful to contrast this immediately with the coproduct. In

$$T + R$$

the two repertoires of operations are merely placed side by side: no new mixed equations are imposed. In

$$T \otimes R$$

they are again both present, but now one adds a systematic interchange law saying that a T -operation may be moved through an R -operation.

For single-output operations

$$f : n \rightarrow 1 \quad \text{in } T, \quad g : m \rightarrow 1 \quad \text{in } R,$$

the characteristic equation says that on an $n \times m$ rectangular array, first applying g along each row and then f gives the same result as first applying f down each column and then g . More generally all operations of the two theories satisfy the corresponding rectangular interchange diagrams.

So the systems reading is

combine two capabilities and require them to act independently/commutatively.

This is a genuinely different architectural choice from coproduct or pushout.

The model-theoretic meaning

The tensor has a particularly clean semantics. For a category $\mathcal{C}$ with finite products, there is a coherent equivalence

$$\text{Mod}(T \otimes R, \mathcal{C}) \simeq \text{Mod}(T, \text{Mod}(R, \mathcal{C})).$$

Thus a $(T \otimes R)$ -model may be read as a T -algebra object in the category of R -algebras, or symmetrically as an R -algebra object in the category of T -algebras.

On a Set-carrier X , this means that X carries both structures and that the T -operations are homomorphisms for the R -structure (equivalently, the R -operations are homomorphisms for the T -structure). This is much stronger than simply having both structures on the same carrier.

The classical Eckmann–Hilton phenomenon is a useful warning about how strong this can be. If T is the Lawvere theory of groups, then

$$T \otimes T$$

is the theory of abelian groups: two group structures satisfying interchange collapse to one commutative group structure.

The tensor product is therefore especially relevant when two kinds of operation are intended to be *independent* in a strong algebraic sense. It should not be used automatically whenever a system has two effects. Many combinations of effects have noncommuting interaction laws, and then a coproduct followed by additional chosen equations, a pushout, a distributive law, or some other construction may be the appropriate object instead.

For our systems programme, the point is that Part III now contains three quite different “combination” operations:

$T + R$: both capabilities, no mixed equations,
 $T \otimes R$: both capabilities, with universal commutation,
 $T_1 \amalg_K T_2$: two capabilities glued along declared common structure.

5 Why completeness and cocompleteness matter

The existence of all these constructions is not decorative. It means that the universe of theories has its own architecture.

For ordinary or suitably ranked Lawvere theories, one can form arbitrary small diagrams

$$D : J \rightarrow \mathbf{Law}$$

and take

$$\lim_J D, \quad \operatorname{colim}_J D.$$

This gives at least three kinds of leverage.

1. **Presentation-independent construction.** We can describe combinations of theories by universal properties rather than repeatedly returning to generators and relations.
2. **Modular system architecture.** Shared subtheories, quotients, and compatibility conditions can themselves be organised into large diagrams and then assembled universally.
3. **Kan extension of indexed families.** Once theories vary over a parameter category, completeness and cocompleteness let us transport whole families by right and left Kan extension, subject to the usual size hypotheses.

The third point will become important below.

6 And what about Poly?

There is a parallel abundance of categorical structure at the polynomial level.

Write a polynomial as

$$p = \sum_{i \in I} y^{E_i}.$$

The category **Poly** of polynomial functors and lenses has all small limits and all small colimits. In fact, its limits agree with the corresponding limits in the ambient functor category $[\mathbf{Set}, \mathbf{Set}]$; the colimits are genuinely polynomial constructions and need not coincide with pointwise colimits of endofunctors.

For limits there is a memorable positional/directional slogan:

positions are combined by limits, while directions are combined contravariantly by colimits.

The general colimit formula is subtler, but the important point for us is simply that the category of interaction shapes itself supports arbitrary small universal constructions.

This means that there are potentially two related categorical calculi available to the systems programme:

universal constructions of algebraic theories in **Law**

and

universal constructions of polynomial interfaces in **Poly**.

6.1 A caution about polynomial monads

We must not overstate what follows.

The theorem

Poly is complete and cocomplete

concerns the category of polynomial functors with lenses. A polynomial monad is additional monoid structure with respect to the composition product. It does *not* follow formally that arbitrary limits or colimits of underlying polynomials automatically carry the required monad structure, or that the corresponding category of polynomial monads creates all such limits and colimits.

So there are two questions:

construct the carrier in **Poly**,

and

lift that construction to monad/theory structure.

The first has a very generous answer; the second needs a separate theorem or construction.

This distinction will matter if we want the polynomial side of the programme to interact systematically with **Run(Poly)**, polynomial monads, or the category of Lawvere theories.

7 Theories that vary with parameters

So far we have written T_{Moore} as though there were a single Moore theory. But the theory actually depends on its input and output sets:

$$T_{\text{Moore}}(A, O).$$

It is generated by

$$\text{step}_a \quad (a \in A)$$

and

$$\text{get} : y^O \rightarrow y,$$

with the reader equations on **get**.

The dependence on A and O has mixed variance.

7.1 Input maps are covariant

Given

$$u : A \rightarrow A',$$

we can send

$$\text{step}_a \mapsto \text{step}_{u(a)}.$$

So the input parameter varies covariantly.

7.2 Output maps are contravariant

Suppose

$$v : O' \rightarrow O.$$

An O -ary get can be interpreted using an O' -ary get by reindexing the family of continuations:

$$\text{get}_O((x_o)_{o \in O}) \mapsto \text{get}_{O'}((x_{v(o')})_{o' \in O'}).$$

So the output parameter varies contravariantly.

Therefore the Moore theories form a diagram

$$T_{\text{Moore}}(-, -) : \mathbf{Set} \times \mathbf{Set}^{op} \longrightarrow \mathbf{Law}.$$

A morphism in the parameter category is a pair

$$(u, v) : (A, O) \longrightarrow (A', O')$$

with

$$u : A \rightarrow A', \quad v : O' \rightarrow O,$$

and it induces a tight theory map

$$T_{\text{Moore}}(A, O) \longrightarrow T_{\text{Moore}}(A', O').$$

7.3 What parameter change does to an actual Moore machine

This mixed variance becomes very intuitive on comodels.

Suppose

$$(S, \delta', o')$$

is an (A', O') -Moore machine:

$$\delta' : A' \times S \rightarrow S, \quad o' : S \rightarrow O'.$$

Restrict it along (u, v) . The resulting (A, O) -machine has

$$\delta(a, s) = \delta'(u(a), s), \quad o(s) = v(o'(s)).$$

Thus the parameter map acts exactly like an interface adapter:

$$\boxed{\text{translate incoming } A\text{-inputs forward along } u, \quad \text{translate outgoing } O'\text{-outputs along } v.}$$

This is a very concrete class of tight maps, but it is only a special class. An arbitrary tight map between Moore theories may send a primitive step to a composite term, or mix operations in ways not induced by a bare function of parameters. Indeed, the parameter category just described captures only the non-dependent reparametrisations of a Moore interface. There is a larger and more natural parameter category already visible from the polynomial description of Moore machines.

Recall that a $T_{\text{Moore}}(A, O)$ -comodel is exactly a coalgebra for the monomial polynomial

$$Oy^A.$$

A general polynomial lens

$$O'y^{A'} \longrightarrow Oy^A$$

between monomials consists of a forward map

$$v : O' \longrightarrow O$$

together with a backwards map

$$u : O' \times A \longrightarrow A'.$$

The crucial point is that the translation of an input $a \in A$ is allowed to depend on the current observation $o' \in O'$.

Such a lens induces a tight interpretation

$$T_{\text{Moore}}(A, O) \longrightarrow T_{\text{Moore}}(A', O').$$

On the reader operation it acts by postcomposing the observed value with

$$v : O' \rightarrow O.$$

On a step operation it acts by first reading the current O' -value and then choosing the corresponding A' -input:

$$\boxed{\text{step}_a \longmapsto \text{get}'(o' \mapsto \text{step}'_{u(o', a)}).$$

Thus the translated input is genuinely state-dependent through the exposed observation.

On comodels, restriction along this tight map is exactly composition with the corresponding polynomial lens. If

$$S \longrightarrow O' \times S^{A'}$$

is a Moore machine with output

$$o' : S \rightarrow O'$$

and transition

$$\delta' : A' \times S \rightarrow S,$$

then the restricted $T_{\text{Moore}}(A, O)$ -comodel has output

$$o = v \circ o'$$

and transition

$$\boxed{\delta(a, s) = \delta'(u(o'(s), a), s).$$

So polynomial lens composition is precisely tight reparametrisation of Moore machines.

The earlier category

$$\mathbf{Set} \times \mathbf{Set}^{\text{op}}$$

is recovered as the special case in which the backwards map is independent of the observation:

$$u(o', a) = \bar{u}(a).$$

Hence it should be regarded as the non-dependent fragment rather than the full parameter category.

In this sense the natural parameter category of Moore theories is the opposite of the monomial fragment of **Poly**:

$$\boxed{\mathcal{P}_{\text{Moore}} \simeq \mathbf{Poly}_{\text{mon}}^{\text{op}},}$$

up to the convention chosen for the direction of polynomial lenses. The family

$$(A, O) \mapsto T_{\text{Moore}}(A, O)$$

therefore extends from ordinary changes of input and output sets to dependent interface transformations.

This is more than a notational enlargement. The polynomial viewpoint says that reparametrising an input port may legitimately depend on what the machine currently exposes at its output port. Tight theory maps already know how to express that dependence:

observe first, then choose the translated input.

The Set-based parameter category sees only the special case in which these two stages decouple.

8 Combining parameterised theories

Once we recognise

$$T_{\text{Moore}} : \mathcal{P} \rightarrow \mathbf{Law}, \quad \mathcal{P} = \mathbf{Set} \times \mathbf{Set}^{op},$$

as a diagram of theories, we can combine the parameter story with the universal constructions of **Law**.

8.1 A span of interfaces gives a span of theories

Suppose in the parameter category we have

$$p_0 \rightarrow p_1, \quad p_0 \rightarrow p_2.$$

Applying T_{Moore} gives

$$T_{\text{Moore}}(p_0) \rightarrow T_{\text{Moore}}(p_1), \quad T_{\text{Moore}}(p_0) \rightarrow T_{\text{Moore}}(p_2).$$

We may therefore form the pushout

$$\boxed{T_{\text{Moore}}(p_1) \amalg_{T_{\text{Moore}}(p_0)} T_{\text{Moore}}(p_2)}.$$

This is an interface-aware gluing of Moore theories: the two theories are combined while the operations identified by the shared parameterised subinterface are forced to agree.

The result need not itself be a Moore theory of the simple form

$$T_{\text{Moore}}(A, O)$$

for some obvious pair (A, O) . That would require a preservation theorem for the functor $T_{\text{Moore}}(-, -)$, and we should not assume it.

This is an important conceptual gain. Universal constructions in **Law** let us leave the original parameterised family when the architecture demands a genuinely new combined theory.

8.2 Parameter arithmetic and theory arithmetic are different

It is tempting to expect formulas such as

$$T_{\text{Moore}}(A_1, O_1) + T_{\text{Moore}}(A_2, O_2) \stackrel{?}{\cong} T_{\text{Moore}}(A_1 + A_2, \text{some expression in } O_1, O_2).$$

Sometimes a useful comparison map may exist, but there is no general reason for such identities to hold. The reader operations and their equations already make the output side nontrivial.

So we should distinguish

combining parameters

from

combining the corresponding theories.

The functor T_{Moore} relates the two, but need not preserve every categorical construction.

8.3 Tensoring parameterised families

The tensor product interacts especially neatly with the parameterised viewpoint. Suppose

$$T : \mathcal{P} \rightarrow \mathbf{Law}, \quad R : \mathcal{Q} \rightarrow \mathbf{Law}$$

are two families of theories. Since $\otimes$ is a bifunctor on $\mathbf{Law}$, there is an induced family

$$T \boxtimes R : \mathcal{P} \times \mathcal{Q} \longrightarrow \mathbf{Law}$$

defined pointwise by

$$(T \boxtimes R)(p, q) = T(p) \otimes R(q).$$

Thus one can vary two interfaces independently while requiring their operations to commute at every parameter value. For example, given a parameterised Moore theory $T_{\text{Moore}}(A, O)$ and some other parameterised theory $R(q)$, the object

$$T_{\text{Moore}}(A, O) \otimes R(q)$$

describes a combined interface carrying both structures with universal interchange between them.

Again this should be distinguished from

$$T_{\text{Moore}}(A, O) + R(q),$$

which merely makes both operation sets available. Parameterisation tells us *which* theories are being combined; the choice of coproduct, tensor, pushout, and so on tells us *how* their operations are meant to interact.

8.4 Kan extension: transporting whole families of theories

Suppose

$$F : \mathcal{P} \rightarrow \mathcal{Q}$$

is a change of parameter scheme and

$$T : \mathcal{P} \rightarrow \mathbf{Law}$$

is an indexed family of theories. Since **Law** is complete and cocomplete, under the usual smallness conditions we can form

$$\mathrm{Lan}_F T \quad \text{and} \quad \mathrm{Ran}_F T.$$

These are universal ways to transport the whole family from $\mathcal{P}$ to $\mathcal{Q}$. Pointwise, the left Kan extension assembles theories by colimits over parameter fibres, while the right Kan extension assembles them by limits.

This is a considerably more sophisticated use of “parameter change” than simply renaming an input alphabet. It suggests that indexed theories can themselves be aggregated, forgotten, or reconstructed along maps of parameter spaces.

9 The total category of Moore-interface theories

The preceding construction has a standard categorical form. The parameterised Moore family

$$T_{\mathrm{Moore}} : \mathcal{P} \longrightarrow \mathbf{Law}$$

determines the comma category

$$T_{\mathrm{Moore}} \downarrow \mathbf{Law}.$$

An object is a parameter

$$p \in \mathcal{P},$$

a theory R , and a tight map

$$j : T_{\mathrm{Moore}}(p) \longrightarrow R.$$

Thus an object may be written

$$(p, j : T_{\mathrm{Moore}}(p) \rightarrow R).$$

We read this as a theory R equipped with a declared Moore interface of type p .

A morphism

$$(p, j : T_{\mathrm{Moore}}(p) \rightarrow R) \longrightarrow (q, k : T_{\mathrm{Moore}}(q) \rightarrow R')$$

consists of a parameter map

$$u : p \longrightarrow q$$

and a tight map

$$h : R \longrightarrow R'$$

such that

$$h \circ j = k \circ T_{\mathrm{Moore}}(u).$$

Equivalently, the square

$$\begin{array}{ccc} T_{\mathrm{Moore}}(p) & \xrightarrow{j} & R \\ T_{\mathrm{Moore}}(u) \downarrow & & \downarrow h \\ T_{\mathrm{Moore}}(q) & \xrightarrow{k} & R' \end{array}$$

commutes.

Thus the total category packages two kinds of change at once:

$$\boxed{\text{change the interface parameters}} \quad \text{and} \quad \boxed{\text{change the implementing theory.}}$$

The evident projection

$$T_{\text{Moore}} \downarrow \mathbf{Law} \longrightarrow \mathcal{P}, \quad (p, j : T_{\text{Moore}}(p) \rightarrow R) \longmapsto p,$$

has fibre over p

$$T_{\text{Moore}}(p) \downarrow \mathbf{Law}.$$

Equivalently, this projection is the Grothendieck construction of the indexed category

$$p \longmapsto T_{\text{Moore}}(p) \downarrow \mathbf{Law},$$

with reindexing along

$$u : p \rightarrow q$$

given by precomposition with

$$T_{\text{Moore}}(u) : T_{\text{Moore}}(p) \rightarrow T_{\text{Moore}}(q).$$

So the indexed-category and comma-category descriptions are two views of the same construction: the former emphasizes fibres and reindexing, while the latter gives the total category immediately.

The choice of coslices reflects a modelling decision. We have taken a theory to *carry* an interface via a map

$$T_{\text{Moore}}(p) \rightarrow R.$$

If instead the intended semantics used theories mapping *to* an interface, one would use slices rather than coslices, with the corresponding change of variance.

10 Tight maps and loose runners in one picture

We can now return to the double category **Run**.

Its objects are theories/monads, its tight arrows are direct theory maps, and its loose arrows are runners. A square has the schematic form

$$\begin{array}{ccc} T & \xRightarrow{Y} & R \\ f \downarrow & & \downarrow g \\ T' & \xRightarrow{Y'} & R' \end{array}$$

where the double-cell data express compatibility between the two runners and the two tight changes of theory.

The interpretation is straightforward even before writing the formal cell axiom:

If we change the abstract theory by f and the residual theory by g , the square witnesses that the implementation Y transforms coherently into the implementation Y' .

This is the place where the two halves of the tutorial finally meet.

10.1 Theory change can happen on either side of a runner

Starting from

$$T \rightsquigarrow R,$$

a tight map on the source side changes what counts as an abstract request. A tight map on the target side changes the residual language in which the implementation is expressed.

For example, a quotient

$$T \twoheadrightarrow T/\sim$$

on the source side strengthens the correctness equations the runner must satisfy. A compilation map

$$R \rightarrow R'$$

on the residual side may lower the residual effects further into another language.

A square says that these changes respect one another.

10.2 Wire first or realise first?

Suppose an external theory is interpreted by a wiring map

$$W : T \rightarrow T_1 + \cdots + T_n,$$

and suppose each component theory T_i is itself realised through some residual theory R_i .

There are then two conceptual routes:

wire the abstract components, then realise the composite,

or

realise the components first, then combine their implementations.

A mature **Run**-based compositional semantics should explain when these routes agree and what comparison cell relates them.

We should not assume that the answer is automatic. Combining runners may require monoidal or duoidal structure, and the coproduct operation on theories must interact appropriately with that structure. But this is exactly the kind of coherence question for which the double-category viewpoint is designed.

The emerging division of labour is:

tight maps organise or reinterpret interfaces

while

loose maps realise interfaces by stateful mechanisms.

The 2-cells of **Run** say when organisation and realisation are compatible.

11 Toward an indexed or fibred Run

The parameterised Moore family suggests one further step.

We have a base of interfaces

$$\mathcal{P} = \mathbf{Set} \times \mathbf{Set}^{op}$$

and a canonical theory map

$$T_{\text{Moore}}(u) : T_{\text{Moore}}(p) \rightarrow T_{\text{Moore}}(q)$$

for every parameter change $u : p \rightarrow q$.

At the tight level, the coslice construction above produced an indexed category

$$\mathcal{F} : \mathcal{P}^{op} \rightarrow \mathbf{Cat}.$$

To include runners as well, we would like an indexed *double* category or a double Grothendieck construction in which:

- objects carry an interface parameter p and an implementing theory;
- vertical/tight arrows may change both the parameter and the theory by a commuting square;
- horizontal/loose arrows are runners between implementing theories;
- 2-cells express compatibility of the runner with the parameter change and the tight interpretation maps.

Schematically, the basic compatibility one would like to organise looks like

$$\begin{array}{ccc} T_{\text{Moore}}(p) & \xRightarrow{Y} & R \\ T_{\text{Moore}}(u) \downarrow & & \downarrow h \\ T_{\text{Moore}}(q) & \xRightarrow{Y'} & R'. \end{array}$$

There is a technical choice hidden here. A parameter map is naturally a tight arrow; if one wants it to participate directly in horizontal composition, one may need companions/conjoints or an equipment-like enhancement. So the precise indexed-double-category construction should be treated as a mathematical problem, not as notation we have already earned.

Still, the target is clear:

interfaces + parameter change + theory interpretation + stateful realisation.

That would put the full systems story in one categorical object.

12 A worked parameter-change example

Before ending, let us make one parameterised example completely explicit.

Take an (A', O') -Moore machine

$$M' = (S, \delta', o').$$

Suppose we want to expose it through a smaller or differently named interface (A, O) . Choose

$$u : A \rightarrow A', \quad v : O' \rightarrow O.$$

The tight map

$$T_{\text{Moore}}(A, O) \longrightarrow T_{\text{Moore}}(A', O')$$

means:

1. when the external world supplies $a \in A$, feed $u(a) \in A'$ to the internal machine;
2. when the internal machine reports $o' \in O'$, expose only $v(o') \in O$.

Thus the externally visible machine is

$$M = (S, \delta, o)$$

with

$$\delta(a, s) = \delta'(u(a), s), \quad o(s) = v(o'(s)).$$

Now suppose M' itself is not a direct comodel but a residual runner

$$T_{\text{Moore}}(A', O') \rightsquigarrow R.$$

Precomposing on the tight source side with

$$T_{\text{Moore}}(A, O) \rightarrow T_{\text{Moore}}(A', O')$$

should give an (A, O) -view of the same implementation. In other words, parameter adaptation is not a separate theory from runners; it is exactly the kind of tight source change that the double category is supposed to accommodate.

This example captures the intended relationship between parameterised theories and **Run** in the simplest possible form.

13 What Part III has added

It is worth collecting the new constructions and their systems readings.

Construction	Systems reading
Tight map $T \rightarrow R$	Direct interpretation: rename, forget, compile, quotient, adapt an interface, or wire operations into target terms.
Coproduct $T + R$	Put two operation repertoires side by side, without mixed equations.
Tensor $T \otimes R$	Put two operation repertoires together and universally require every operation of one to commute with every operation of the other.
Coequalizer	Impose equations universally; turn raw operations into a lawful quotient theory.
Pushout $T_1 \amalg_K T_2$	Assemble theories while sharing or identifying a specified common subtheory.
Equalizer	Isolate the universal fragment on which two interpretations agree.
Product/pullback	Organise pairs of target interpretations, with optional compatibility through a common target.
Parameter map	Adapt an interface by translating inputs and outputs; gives a distinguished class of tight maps.
Indexed coslices	Package theories equipped with a declared parameterised interface and allow simultaneous parameter/theory change.
Squares in Run	Express compatibility between direct theory change and stateful realisation.

Three distinctions now seem especially important.

1. **Wiring is not all of tightness.** A wiring map is a theory interpretation with a particular systems reading; tight maps also express compilation, restriction, quotienting, and parameter adaptation.
2. **Parameter combinations are not automatically theory combinations.** A family $T : \mathcal{P} \rightarrow \mathbf{Law}$ relates the two levels, but limits and colimits in $\mathcal{P}$ need not be preserved by T . The category **Law** supplies genuinely new combined theories when needed, and its tensor product supplies a further, non-(co)limit way of demanding universal commutation between operation families.
3. **A diagram is not yet a fibration.** The family $T_{\text{Moore}} : \mathcal{P} \rightarrow \mathbf{Law}$ records canonical change-of-parameter maps. To obtain an indexed category one chooses fibres, such as $T_{\text{Moore}}(p) \downarrow \mathbf{Law}$, and only then applies the Grothendieck construction.

14 Where to go next

Parts I–III now give us three layers:

Part I: comodels and ordinary machines as runners into Id,
 Part II: residual runners and machine-on-machine realisation,
 Part III: tight interpretations, universal theory-building, and parameter change.

The main philosophical picture has become richer than “patterns run on matter.” We now have two orthogonal ways of moving between theories:

tight interpretation changes the language; loose realisation supplies a mechanism.

Universal constructions tell us how languages themselves can be assembled, while parameterisation tells us how whole families of interfaces vary. The double-category viewpoint is what promises to make all of these changes compositional at once.

References and orientation

For the tight/loose structure and the double category of runners, the standing reference remains:

- D. McDermott, E. Rivas, and T. Uustalu, *Sweedler Theory of Monads*, FoSSaCS 2022, LNCS 13242, 428–448.

For completeness and cocompleteness of categories of Lawvere theories, a useful reference is:

- O. Kammar, *An Algebraic Theory of Type-and-Effect Systems*, DPhil thesis, University of Edinburgh. In the enriched/ranked setting, the category of Lawvere theories is locally presentable; Theorems 6.16–6.17 record cocompleteness, completeness, and the componentwise description of limits. For the countable case, see also M. Hyland, G. Plotkin, and J. Power, work on combining effects and Lawvere theories.

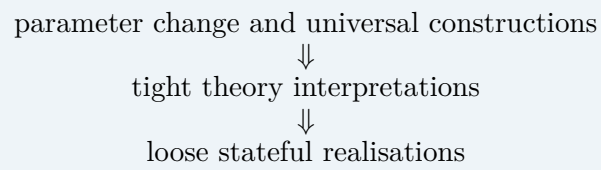
For the tensor product of Lawvere theories and its model-theoretic interpretation:

- M. Hyland and J. Power, *The Category Theoretic Understanding of Universal Algebra: Lawvere Theories and Monads*. Their Definition 3.2 gives the Kronecker tensor by universal commutation of operations; Proposition 3.3 makes **Law** symmetric monoidal under $\otimes$; and Theorem 3.4 identifies models of $T \otimes R$ with T -models internal to R -models. The same discussion recovers the Eckmann–Hilton example that the tensor square of the group theory is the theory of abelian groups.

For **Poly**:

- N. Niu and D. I. Spivak, *Polynomial Functors: A Mathematical Theory of Interaction*, Chapter 5. Theorem 5.33 proves that **Poly** has all small limits, and Theorem 5.43 proves that it has all small colimits. Limits agree with those in **[Set, Set]**, while general colimits need not.

The picture to retain from Part III is



with the eventual goal of organising all three directions in one fibred double-categorical systems semantics.