

# Getting to Grips with Sweedler Runners for Systems

A conversational tutorial, Part VIII: process kernels, graphical species, and Petri nets  
From many-input/many-output processes to higher-dimensional concurrency

September 2026

**Aim.** In the preceding Parts our favourite ambient categories were built from endofunctors: a theory acted on a space of states, and a runner explained how one theory could be realised in another. Petri nets force a useful change of viewpoint. A process may have many resources at its input and many at its output, and independent processes may happen in parallel. We therefore begin not with Petri nets themselves but with a category of *typed process kernels*

$$\mathcal{D}_S = [\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}].$$

An object of  $\mathcal{D}_S$  tells us what processes can carry one finite  $S$ -typed resource boundary to another. For fixed palette  $S$ , this is also a familiar category of  $\Sigma$ -net data: the point is not to deny that Petri-style nets live here, but to recognise that the same bicollection can be read as a typed process kernel before any free execution semantics is imposed. This lets us ask what else can be wired to a net, how changing token symmetry changes the base itself, and how the resulting process semantics should compare with the HDA semantics of Part VII.

## Contents

|          |                                                                              |          |
|----------|------------------------------------------------------------------------------|----------|
| <b>1</b> | <b>The change of viewpoint: from operations on states to process kernels</b> | <b>2</b> |
| 1.1      | A few process kernels before Petri nets . . . . .                            | 3        |
| <b>2</b> | <b>Two compositions: series and parallel</b>                                 | <b>4</b> |
| 2.1      | Sequential composition by coend . . . . .                                    | 4        |
| 2.2      | Parallel composition by Day convolution . . . . .                            | 4        |
| <b>3</b> | <b>Why a presheaf category still belongs to the endofunctor story</b>        | <b>5</b> |
| <b>4</b> | <b>Why symmetry appears</b>                                                  | <b>5</b> |
| <b>5</b> | <b>From bicollections to digraphical species</b>                             | <b>6</b> |
| <b>6</b> | <b>Petri nets enter as a particularly useful test family</b>                 | <b>7</b> |
| 6.1      | A tiny example . . . . .                                                     | 7        |
| <b>7</b> | <b>Three bases: ordered, symmetric and collective tokens</b>                 | <b>7</b> |

|           |                                                                  |           |
|-----------|------------------------------------------------------------------|-----------|
| <b>8</b>  | <b>What can be wired to a Petri process theory?</b>              | <b>8</b>  |
| 8.1       | Implementation and refinement . . . . .                          | 8         |
| 8.2       | A controller or regulator . . . . .                              | 8         |
| 8.3       | An observer, monitor or certifier . . . . .                      | 9         |
| 8.4       | A stochastic, timed or quantitative environment . . . . .        | 9         |
| <b>9</b>  | <b>How far does the basic Petri picture stretch?</b>             | <b>9</b>  |
| 9.1       | Two meanings of colour . . . . .                                 | 10        |
| 9.2       | Why inhibitor and reset arcs are diagnostic . . . . .            | 10        |
| <b>10</b> | <b>Open Petri nets: a second kind of boundary</b>                | <b>11</b> |
| <b>11</b> | <b>From Petri nets to higher-dimensional automata</b>            | <b>12</b> |
| <b>12</b> | <b>Where should Run live?</b>                                    | <b>12</b> |
| 12.1      | Sequential theories first . . . . .                              | 13        |
| <b>13</b> | <b>What should a runner mean here?</b>                           | <b>13</b> |
| <b>14</b> | <b>The palette should vary</b>                                   | <b>14</b> |
| <b>15</b> | <b>What has been established, and what is now genuinely open</b> | <b>14</b> |
| <b>16</b> | <b>The larger lesson</b>                                         | <b>16</b> |
|           | <b>References</b>                                                | <b>16</b> |

## 1 The change of viewpoint: from operations on states to process kernels

Up to now we have often started from an endofunctor category such as

$$[\mathbf{Set}, \mathbf{Set}]_{\text{acc}}$$

or one of its enriched, graded, smooth or cubical relatives. An object there takes a state-like object  $X$  and produces another state-like object. That is exactly the right language for effects and operations with one ambient carrier.

Petri-style systems suggest a different first picture. A transition may consume several resources and produce several resources. If  $S$  is a set of resource types, let  $\mathbb{B}_S$  denote the groupoid of finite  $S$ -labelled sets and label-preserving bijections. Think of an object

$$A \in \mathbb{B}_S$$

as a finite typed boundary: a finite collection of resource occurrences, each carrying a type in  $S$ .

Now consider

$$\mathcal{D}_S = [\mathbb{B}_S^{\text{op}} \times \mathbb{B}_S, \mathbf{Set}].$$

An object  $P \in \mathcal{D}_S$  assigns a set  $P(A, B)$  to each pair of finite typed boundaries. We read an element of  $P(A, B)$  as a process

$$A \longrightarrow B.$$

Naturality says that renaming the individual occurrences in  $A$  and  $B$  changes only the presentation, not the underlying process.

**The first conceptual shift.**  $\mathcal{D}_S$  should not be introduced as “the category in which Petri nets live”. It is a category of *typed many-input/many-output process kernels*. Petri transitions are one important family of generators for such kernels.

### 1.1 A few process kernels before Petri nets

It helps to see several examples before mentioning places and transitions.

**The wire kernel.** The hom-profunctor

$$I(A, B) = \mathbb{B}_S(A, B)$$

contains only reversible relabellings of a boundary. It is the process that does no work apart from carrying resources through unchanged.

**A resource converter.** Take resource types  $a, b, c \in S$ . We may specify a generator

$$\alpha : 2a + b \longrightarrow c.$$

Here  $2a + b$  is shorthand for a finite  $S$ -labelled set containing two occurrences of  $a$  and one of  $b$ . Nothing yet says that  $\alpha$  is a Petri transition. It might be a chemical reaction, a manufacturing step, a proof rule, a circuit component, or an abstract resource conversion.

**A reversible component.** We might have generators

$$\rho : a + b \longrightarrow c + d, \quad \rho^{-1} : c + d \longrightarrow a + b,$$

together with equations saying that the two composites are identities. This looks more like a reversible circuit or groupoidal process component than an ordinary Petri transition.

**A rewriting kernel.** Let the types in  $S$  be symbols or object sorts. A collection of rewrite rules

$$A \rightsquigarrow B$$

again gives many-input/many-output process generators. The same boundary calculus can be read computationally, chemically, logically, or operationally.

**A quantitative kernel.** Instead of merely asking whether a process  $A \rightarrow B$  exists, one may later want costs, probabilities, rates, durations, predicates or other values attached to it. That may suggest replacing **Set**, enriching or grading the kernel category, or keeping a qualitative source theory and mapping it into a quantitative target theory.

The moral is that the boundary language already belongs to a broad process calculus. Petri nets become interesting because they supply a particularly economical local syntax for resource transformations and a rich theory of their executions.

## 2 Two compositions: series and parallel

The reason  $\mathcal{D}_S$  is attractive for our Sweedler programme is that it carries two compositions with immediate systems meanings.

### 2.1 Sequential composition by coend

For endoprofunctors  $P, Q \in \mathcal{D}_S$ , define

$$(P \odot Q)(A, C) = \int^{B \in \mathbb{B}_S} P(A, B) \times Q(B, C).$$

An element is represented by a middle boundary  $B$ , a  $P$ -process  $A \rightarrow B$ , and a  $Q$ -process  $B \rightarrow C$ , with the coend identifying coordinate changes on  $B$ . Thus

$$\boxed{\odot = \text{series composition.}}$$

The unit is the wire kernel  $I = \mathbb{B}_S(-, -)$ . A monoid for  $\odot$  is a promonad on  $\mathbb{B}_S$ : a category-like theory of processes whose objects are typed resource boundaries and whose arrows include their symmetries.

### 2.2 Parallel composition by Day convolution

The groupoid  $\mathbb{B}_S$  is symmetric monoidal under disjoint union. Day convolution gives a second tensor. Schematically,

$$\begin{aligned} (P \boxtimes Q)(A, B) &= \int^{A_1, A_2, B_1, B_2} \mathbb{B}_S(A, A_1 \sqcup A_2) \\ &\quad \times P(A_1, B_1) \times Q(A_2, B_2) \times \mathbb{B}_S(B_1 \sqcup B_2, B). \end{aligned}$$

This says: split the incoming resources, run two processes independently, and reunite their outgoing resources. Hence

$$\boxed{\boxtimes = \text{parallel juxtaposition.}}$$

The two products interact by the canonical interchange map

$$(P \odot P') \boxtimes (Q \odot Q') \longrightarrow (P \boxtimes Q) \odot (P' \boxtimes Q').$$

Two sequential composites can always be placed in parallel: their middle boundaries combine as  $B_1 \sqcup B_2$ . The converse direction is not canonical, since a middle boundary appearing in a composite of parallel kernels may admit incompatible splittings at the two stages. This is the same orientation as composition together with Day convolution in the endofunctor examples of Part IV [6].

**Monoid or prop?** A monoid for  $\odot$  has sequential composition. A prop also has a compatible parallel tensor. So a prop is not merely an arbitrary object of  $\mathbf{Run}(\mathcal{D}_S)$  if  $\mathbf{Run}$ -objects are taken to be  $\odot$ -monoids. It is closer to a suitably structured monoid, or duoid-like object, in the two-fold monoidal setting. We shall keep this distinction visible rather than decide too early which level should be primary.

### 3 Why a presheaf category still belongs to the endofunctor story

At first sight Part VIII may feel more like Part VI's  $C\text{-Set}$  than its endofunctor construction. An object of  $\mathcal{D}_S$  is indexed over input and output profiles; it looks like structured data spread over a base. That impression is useful, but it is only half the story.

An endoprofunctor is also a *kernel*. Let

$$\widehat{\mathbb{B}}_S = [\mathbb{B}_S^{op}, \mathbf{Set}]$$

be the presheaf category of  $S$ -typed finite-resource species. A kernel

$$P : \mathbb{B}_S^{op} \times \mathbb{B}_S \longrightarrow \mathbf{Set}$$

acts on presheaves by a coend. With one conventional choice of variance,

$$(\widehat{P}X)(B) = \int^{A \in \mathbb{B}_S} X(A) \times P(A, B).$$

Thus  $P$  determines a cocontinuous endofunctor

$$\widehat{P} : \widehat{\mathbb{B}}_S \longrightarrow \widehat{\mathbb{B}}_S.$$

Conversely, the free-cocompletion property of a presheaf category says that cocontinuous endofunctors are determined by such kernels. Up to the chosen variance convention, one has the familiar equivalence

$$[\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}] \simeq \text{Cocont}([\mathbb{B}_S, \mathbf{Set}], [\mathbb{B}_S, \mathbf{Set}]),$$

**Presheaf-like and endofunctor-like at once.**

$$\mathcal{D}_S = [\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}] \simeq \text{Cocont}([\mathbb{B}_S, \mathbf{Set}], [\mathbb{B}_S, \mathbf{Set}]).$$

The left-hand description makes boundaries and symmetries explicit. The right-hand description reconnects us with the endofunctor architecture used throughout the tutorial.

This sharpens the comparison with Part VI. An object of  $C\text{-Set}$  is graded over one resource index. Here we have two variances, input and output, so the indexing data already behaves as a process transformer. The category has the *feel* of a presheaf category while retaining an endofunctor interpretation.

### 4 Why symmetry appears

Why is  $\mathbb{B}_S$  a groupoid rather than merely a set of multiplicity vectors? Because resource boundaries carry permutation symmetry.

A polynomial functor

$$P(X) = \sum_{b \in B} X^{E_b}$$

has rigidly visible input positions. Species relax this. Let  $\mathbb{B}$  be the groupoid of finite sets and bijections. A combinatorial species is a functor

$$F : \mathbb{B} \longrightarrow \mathbf{Set},$$

with associated analytic functor

$$\widehat{F}(X) = \coprod_{n \geq 0} F[n] \times_{S_n} X^n.$$

The bag species gives

$$\text{Bag}(X) = \coprod_{n \geq 0} X^n / S_n.$$

The quotient records that relabellings of the finite input set are coordinate changes rather than new operations [1, 2].

The same principle now operates on both sides of a process. A value  $P(A, B)$  varies naturally under bijections of  $A$  and  $B$ . Permutation symmetry is therefore not nuisance bookkeeping; it is part of the system type.

## 5 From bicollextions to digraphical species

A kernel indexed by two boundaries still tells us only the type  $A \rightarrow B$  of a local process. To describe composites whose internal shape is an arbitrary graph, we need graphical species.

Kock's elementary directed graphs consist of the unit edge  $U$  and the corollas  $C_n^m$ , one node with  $m$  imports and  $n$  exports. Let

$$\mathbf{elGr} \subset \mathbf{Gr}$$

be the category of elementary graphs and etale maps. A digraphical species is a presheaf

$$F : \mathbf{elGr}^{op} \longrightarrow \mathbf{Set}.$$

Unpacked, it provides

$$F[U] = \text{colours}$$

and sets

$$F[C_n^m]$$

of  $(m, n)$ -ary operations, with the expected actions of  $S_m \times S_n$  and compatibility of edge colours.

|                  | ordinary species            | digraphical species                     |
|------------------|-----------------------------|-----------------------------------------|
| local profile    | $n$ inputs                  | $m$ inputs, $n$ outputs                 |
| symmetry         | $S_n$                       | $S_m \times S_n$                        |
| composite shapes | trees/operadic substitution | directed graphs/properadic substitution |

Connected acyclic graph substitution generates properadic composition; allowing disconnected graphs supplies the independent parallel components expected in prop-like process theories [3, 4]. This is the graphical counterpart of the passage from local operations to a whole theory of composites.

## 6 Petri nets enter as a particularly useful test family

We can now introduce Petri nets without making them define the ambient category.

A whole-grain Petri net is presented by a diagram

$$S \longleftarrow I \longrightarrow T \longleftarrow O \longrightarrow S,$$

where  $S$  is the set of places,  $T$  the set of transitions, and the fibres of  $I \rightarrow T$  and  $O \rightarrow T$  are the individual input and output arcs of each transition [4]. Unlike the usual multiset presentation, a transition consuming two tokens from the same place retains two distinct input arcs, without imposing an order on them.

The restricted Yoneda construction sends such a net into digraphical species. At an  $(m, n)$ -corolla it remembers a transition together with bijections numbering its input and output arcs. The action of  $S_m \times S_n$  changes those numberings. For a whole-grain net that action is free; in Kock's finite setting the image is characterised by this flatness together with finiteness conditions [4].

### 6.1 A tiny example

Take two places  $a, b$  and a transition

$$t : 2a \longrightarrow b.$$

In collective-token notation the input is just the multiplicity  $2a$ . In the whole-grain presentation there are two actual input arcs

$$i_1, i_2 \in I_t$$

both incident with  $a$ . The associated species sees, at the  $(2, 1)$ -corolla, the possible bijections

$$[2] \xrightarrow{\sim} \{i_1, i_2\}.$$

There are two, and the nontrivial element of  $S_2$  swaps them freely.

Thus we should distinguish

$$\begin{array}{ll} 2a & : \text{ collective multiplicity,} \\ \{i_1, i_2\} \rightarrow a & : \text{ coordinate-free individual arcs,} \\ [2] \xrightarrow{\sim} \{i_1, i_2\} & : \text{ one numbered presentation.} \end{array}$$

The species retains all coordinate presentations and records their symmetry rather than choosing one.

## 7 Three bases: ordered, symmetric and collective tokens

The review of token symmetry becomes cleaner if we vary not only the palette but the *boundary category*. For a fixed set of places  $S$  there are three basic bases:

$$S^* \longrightarrow \mathbb{B}_S \xrightarrow{\pi_0} \mathbb{N}[S].$$

Here  $S^*$  is the free strict monoidal category on  $S$  viewed discretely,  $\mathbb{B}_S$  is the free symmetric strict monoidal groupoid, and  $\mathbb{N}[S]$  is the free commutative monoid on  $S$  viewed as a discrete category. Correspondingly one obtains kernel categories

$$[(S^*)^{op} \times S^*, \mathbf{Set}], \quad [\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}], \quad [\mathbb{N}[S]^{op} \times \mathbb{N}[S], \mathbf{Set}].$$

Baez–Genovese–Master–Shulman organise pre-nets,  $\Sigma$ -nets and ordinary collective-token Petri nets around exactly this ordered/symmetric/commutative progression, with free strict monoidal, free symmetric strict monoidal and free commutative monoidal execution semantics respectively [5].

Thus collective-token semantics is not merely a special object of the same  $\mathcal{D}_S$ : it is obtained by changing the base. Whole-grain nets sit inside the  $\Sigma$ -net world as the free-action case, while general  $\Sigma$ -nets allow isotropy. The maps between the three bases therefore give a precise version of the symmetry-forgetting question we shall return to later.

This perspective also points beyond these three examples. Master’s  $Q$ -nets replace the chosen algebra of markings by a Lawvere theory  $Q$ , making change of resource algebra functorial [10]. For a tutorial whose larger programme is built around Lawvere theories and monads, this is a particularly natural neighbouring construction.

## 8 What can be wired to a Petri process theory?

The broad kernel viewpoint immediately provides more examples than “a Petri net running on another Petri net”. Here are four families that deserve to be treated as genuine examples rather than as a final list of possibilities.

### 8.1 Implementation and refinement

Suppose  $T$  is generated by a high-level transition

$$\tau : a + b \longrightarrow c.$$

A lower-level theory  $R$  may realise this by a composite

$$a + b \longrightarrow x + y \longrightarrow z \longrightarrow c.$$

The intermediate resource types  $x, y, z$  are implementation detail. A runner should express the idea that a visible  $T$ -step is executed by an  $R$ -process while residual implementation state or resources are carried along.

This is perhaps the most literal Petri analogue of our earlier examples: one theory is the program or specification, another is the machine on which it runs.

### 8.2 A controller or regulator

A plant may be presented as a process theory generated by Petri transitions, while another component determines which transitions are currently enabled or which resource configurations are admissible. More sophisticated controllers may inspect additional state, priorities, data values, or predicates.

This reconnects with the Moore/Mealy and good-regulator discussion of Parts III and V. There is, however, a qualification. A runner realises each operation of its source theory, whereas a supervisory controller typically *disables* some plant transitions. Such control is naturally partial: it may be represented by a tight subtheory/restriction, or by a loose realisation into a target carrying failure, rather than by an ordinary total runner without further structure.

### 8.3 An observer, monitor or certifier

A process may be wired to something that records

- accumulated cost,
- provenance or causal history,
- a safety invariant,
- energy or resource use,
- transition counts,
- a temporal certificate.

This is the process-theoretic analogue of the predicate and grading constructions from Parts V and VI. The second component may be passive in the sense that it does not determine the visible transition, yet it changes along with the run and can later be queried. If such values are to combine under independent parallel composition, their accumulation must itself be compatible with  $\boxtimes$ ; the basic case is a commutative monoid of costs, weights or provenance values.

### 8.4 A stochastic, timed or quantitative environment

A qualitative process theory can also be realised in a theory carrying firing rates, delay information, probabilities or costs. There are two conceptually different ways to organise this.

First, change the ambient value category so that the process kernel itself is enriched or graded. Second, retain an ordinary qualitative source theory  $T$  and let it run in a quantitative target theory  $R$ . The latter fits particularly well with the residual-theory perspective of Part II: the extra structure need not be built into the syntax of  $T$  if it belongs to the environment in which  $T$  is executed.

**The point of the examples.** Once  $\mathcal{D}_S$  is read as a process-kernel category, a Petri net can be wired to an implementation, a controller, an observer, or a quantitative environment. “Petri-to-Petri” is only one corner of the picture.

## 9 How far does the basic Petri picture stretch?

The phrase “Petri nets” covers many formalisms. A long historical catalogue is not very helpful here. For our purposes it is better to ask *which structural ingredient has changed*. The variants then fall along several largely independent axes.

| axis              | examples                                                    | what the Run setting must notice                                                                             |
|-------------------|-------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| boundary symmetry | pre-nets, whole-grain nets, $\Sigma$ -nets, collective nets | whether transition presentations carry free actions, isotropy, or complete quotienting by token permutations |
| resource typing   | place-coloured or typed nets                                | replace a single untyped boundary groupoid by an indexed palette $S$                                         |

| axis                    | examples                                     | what the Run setting must notice                                                                                |
|-------------------------|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| token data              | high-level/coloured Petri nets               | tokens carry values from colour sets; variables and arc expressions act on those values                         |
| enabling tests          | guards, read/test arcs, inhibitor arcs       | firing depends on information about the ambient marking, not merely on a fixed consumed/produced boundary       |
| non-local updates       | reset, transfer, self-modifying arcs         | a transition may transform an unknown quantity or modify the update rule itself                                 |
| quantitative decoration | weights, costs, rewards                      | enrich or grade process values, or map into a quantitative target theory                                        |
| time                    | timed nets, deterministic or interval delays | add clocks, durations or temporal constraints to process execution                                              |
| probability/rates       | stochastic nets, GSPNs                       | firing carries rate or distributional data and conflict resolution acquires stochastic semantics                |
| state domain            | continuous and hybrid nets                   | markings need no longer live in a discrete multiset domain                                                      |
| openness                | designated input/output places               | distinguish internal transition boundaries from the external interface through which whole systems are composed |

The table suggests a hierarchy rather than a single enormous category of “all Petri nets”. Some variants remain comfortably inside the same boundary-kernel format. Some require a richer value category. Others force us to make the current marking part of what a transition may inspect. Still others change the marking object itself.

## 9.1 Two meanings of colour

There is a particularly important terminological trap. The set  $S$  in  $\mathbb{B}_S$  colours *resource occurrences or places*: it tells us which type of wire a boundary point carries. This is not yet the full apparatus of a high-level coloured Petri net. In the latter, tokens themselves carry data values from colour sets; transitions may contain typed variables, guards and arc expressions [7].

So we should distinguish

typed boundary colour  $\neq$  token data in a coloured Petri net.

The latter is richer and points naturally toward indexed families, dependent typing, or an internal language in which transition rules can mention token data.

## 9.2 Why inhibitor and reset arcs are diagnostic

Ordinary Petri transitions are local in a strong sense. A transition carries finite pre- and post-multisets

$$A \longrightarrow B.$$

An inhibitor arc changes the story: firing may require that some place contain no token, or fewer than a threshold. A read arc may require a resource to be present but leave it untouched. A reset arc may delete however many tokens happen to be present.

These are not merely more elaborate choices of  $A$  and  $B$ . They ask a transition to inspect or transform the *ambient marking*. This exposes a useful boundary of the simple kernel picture:

$$\boxed{\begin{array}{c} \text{fixed local boundary transformation} \\ A \longrightarrow B \end{array}} \quad \rightsquigarrow \quad \boxed{\text{marking-dependent process action.}}$$

The important distinction is not kernel versus endofunctor: those are equivalent presentations of the same cocontinuous data, so moving between them cannot add expressive power. What fails is  $\boxtimes$ -*locality*, or the frame property. For an ordinary local generator  $G$ , a step in a larger marking is obtained schematically as

$$G \boxtimes I,$$

placing the active transition in parallel with untouched context. An inhibitor transition may inspect that context, and a reset transition may modify an amount of context not fixed in advance. Their one-step behaviour can still be encoded by a global kernel indexed by whole markings, hence still as an object of the kernel category, but it need not factor through local generators and idle wires.

This separates two levels. Such globally marking-indexed step kernels can still generate sequential promonads, so they remain available at the sequential level of the theory. What generally fails is presentation by a local prop-like syntax generated under  $\boxtimes$ . If genuinely non-cocontinuous dependence on a marking is required, then one must leave Cocont itself, for example for a suitable category of accessible endofunctors.

## 10 Open Petri nets: a second kind of boundary

We should clearly distinguish the following: A transition boundary

$$A \longrightarrow B$$

is not the same thing as the external interface of an entire open system.

Baez and Master define an open Petri net by designating input and output places through a cospan. Such systems compose by gluing the outputs of one to the inputs of another, and disjoint union gives a symmetric monoidal structure; the construction is naturally organised as a symmetric monoidal double category [8].

Schematically we should distinguish

$$\begin{array}{ll} \text{boundary of a process step} & A \longrightarrow B, \\ \text{interface of a whole open net} & X \longrightarrow P \longleftarrow Y. \end{array}$$

The first describes resources consumed and produced by a process. The second describes the ports through which an entire system is available for external wiring.

This makes open Petri nets especially relevant to the present tutorial. They already inhabit a double-categorical wiring story, while  $\mathbf{Run}(\mathcal{D})$  gives us another double-categorical organisation: tight maps of theories, loose runners, and 2-cells between runner squares. The relationship between the two should not be assumed to be identity. Rather, it is a comparison problem.

Structured cospans provide a wider common language for open networks, including Petri nets, electrical circuits and chemical reaction networks [9]. So this is another answer to our opening question “what can be wired to a Petri net?”: at the external-interface level, a Petri component can participate in the same compositional architecture as other network formalisms once a suitable common interface semantics has been chosen.

## 11 From Petri nets to higher-dimensional automata

Part VII represented true concurrency cubically, using higher-dimensional cells to witness independent actions. The graphical semantics developed here represents true concurrency by process graphs and partial-order-like execution shapes. These are not rival descriptions that must collapse into one another; they are two geometries of concurrency that should be compared.

There is already a concrete bridge. The Petri-to-HDA semantics going back to van Glabbeek has recently been revisited by Amrane, Bazille, Fahrenberg, Helouet and Schlehuber-Caissier. Their construction treats ordinary Petri nets and extends the translation to inhibitor arcs and generalized self-modifying nets [13].

There are, however, two different corners to compare. The HDA constructed from a chosen marking is a state-space semantics: its cells describe executions reachable from that marking. It should therefore not be displayed as though a process theory  $T$  were simply sent to another theory  $H(T)$ . We should instead distinguish

$$\begin{aligned} \text{kernel/net syntax} &\longrightarrow \text{cubical process theory,} \\ \text{chosen marking and its executions} &\longrightarrow \text{cubical state/HDA.} \end{aligned}$$

The second line is the one directly suggested by the known Petri-to-HDA construction. The first would be the theory-level comparison needed before claiming a functor between Run worlds. We leave both functoriality and the required duoidal compatibility as research questions rather than promote the object-level construction prematurely.

The inhibitor-arc case is especially suggestive. In Section 9 inhibitor arcs forced us beyond a purely local  $A \rightarrow B$  view because enabling depends on the ambient marking. Yet the HDA translation can still accommodate them. This makes inhibitor nets a good test case for any proposed comparison between the process-kernel and cubical Run settings.

## 12 Where should Run live?

We can now return to the Sweedler question with a better sense of the surrounding territory.

The presheaf category

$$\mathbf{PSh}(\mathbf{elGr}) = \mathbf{Set}^{\mathbf{elGr}^{op}}$$

is an excellent home for local graphical signatures: colours, transitions, symmetries and properadic or propic substitution all live nearby. On the other hand,

$$\mathcal{D}_S = [\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}]$$

exhibits series and parallel process composition extremely explicitly, and it is simultaneously a kernel category and a category of cocontinuous endofunctors.

For the present tutorial it is useful to divide the labour:

|                               |                                                                                                                      |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------|
| $\mathbf{PSh}(\mathbf{elGr})$ | local graph-shaped signatures, isotropy, free properad/prop constructions                                            |
| $\mathcal{D}_S$               | explicit process kernels with series/parallel duoidal structure and a direct connection to cocontinuous endofunctors |

A future theorem should compare these presentations rather than forcing us to choose one as the unique “correct” Petri setting.

### 12.1 Sequential theories first

The unit calculation above makes the choice less symmetrical than it first appeared. For the ordinary Sweedler construction on  $\odot$ -monoids, a measuring residual  $C$  must carry a comonoid structure for  $\odot$ : in other words,  $C$  is a *procomonad* on  $\mathbb{B}_S$ . Its comultiplication is exactly what is needed to pass through the interchange map when measuring a product in  $T$ .

The counit

$$C \longrightarrow I = \mathbb{B}_S(-, -)$$

already has a strong consequence. If  $A$  and  $B$  are not isomorphic typed boundaries, then  $I(A, B) = \emptyset$ , hence

$$C(A, B) = \emptyset.$$

A residual procomonad can therefore carry hidden information only along type-preserving boundary profiles. This gives a precise sense in which residual state here may be resource-shaped without itself changing the visible resource type.

By contrast, if we insist that the measuring residual for a prop-like or duoid-level theory also be a comonoid for  $\boxtimes$ , its counit must map to the Day unit  $J$ . Since  $J$  is empty away from  $(\emptyset, \emptyset)$ , this forces  $C$  to be concentrated at the empty boundary. Combined with the  $\odot$ -counit, the residual has collapsed to essentially ordinary set-indexed information rather than boundary-threaded state.

**Decision for this Part.** We therefore take the primary Run-objects to be  $\odot$ -monoids (promonads), with  $\boxtimes$  retained as the external parallel tensor. The free prop/SSMC generated by a net remains essential process semantics, but a naively two-sided Sweedler measuring theory of duoids is too restrictive in this non-normal duoidal category. A satisfactory prop-level runner would need a weaker one-sided notion of measuring or a modified ambient base.

This turns the main existence question into a more concrete structural one: understand procomonads on the boundary groupoid  $\mathbb{B}_S$ . A tempting conjecture is that, for a groupoid  $G$ , many such comonoids are of the form

$$C(A, B) \cong X(A) \times G(A, B)$$

for a functor  $X : G \rightarrow \mathbf{Set}$ . We do *not* use this as a theorem here; even the one-object groupoid cases provide useful tests for the proposed classification.

## 13 What should a runner mean here?

Suppose  $T$  and  $R$  are  $\odot$ -monoids in  $\mathcal{D}_S$ . A measuring object

$$\mathcal{M}_S(T, R)$$

should classify ways of realising  $T$ -processes in an  $R$ -environment, with residual structure carried by a procomonad on  $\mathbb{B}_S$ . The general existence technology for measuring comonoids in closed monoidal settings is well established [12]; here the new calculation is not merely existence but the concrete form of the residual procomonads allowed by this process-kernel base.

The hidden object is therefore not automatically a plain set of machine states. It is boundary-indexed and, by the counit argument, type-preserving. This is a sharper statement than the earlier vague claim that the residual might be “resource-shaped”. The examples from Section 8 suggest concrete targets: implementation/refinement; monitoring with cost or provenance; and qualitative processes realised in timed, stochastic or otherwise quantitative targets. Supervisory control requires the partiality qualification already noted in Section 8.

These calculations would do more to establish the usefulness of  $\mathbf{Run}(\mathcal{D}_S)$  than an abstract existence theorem alone.

## 14 The palette should vary

A fixed set  $S$  was convenient for defining  $\mathcal{D}_S$ , but a general systems theory should not freeze the palette forever. A map

$$f : S \rightarrow S'$$

induces a functor

$$\mathbb{B}_S \longrightarrow \mathbb{B}_{S'}$$

and hence restriction and Kan-extension operations between the corresponding presheaf and profunctor categories.

This returns us to Part III. The family

$$S \longmapsto \mathbf{Run}(\mathcal{D}_S)$$

should probably be organised fibredly or indexed over palettes. Changing the resource types is then base change; changing the theory within a palette is a tight map; and a loose arrow is a runner between theories, perhaps after suitable reindexing.

Thus the Petri case reinforces a lesson already encountered with Moore interfaces and graded bases:

parameters belong in the base; theories and runners live over them.

High-level coloured Petri nets suggest that even this may eventually need refinement. Once token values live in nontrivial colour sets and transitions carry variables and guards, the base may have to remember more than a bare set of place names.

## 15 What has been established, and what is now genuinely open

We can separate background mathematics from the new Run questions:

### Established background

- Species and analytic functors organise permutation symmetries of finite inputs [1, 2].
- Digraphical species extend this to many-input/many-output graphical operations and graph substitution [3].
- Whole-grain Petri nets embed as flat digraphical species, while general  $\Sigma$ -nets permit arbitrary isotropy [4, 5].

- Open Petri nets form a compositional double-categorical wiring formalism using designated external interfaces [8].
- There are established Petri-to-HDA translations, including extensions to inhibitor arcs and generalized self-modifying nets [13].

### Well-motivated Run programme

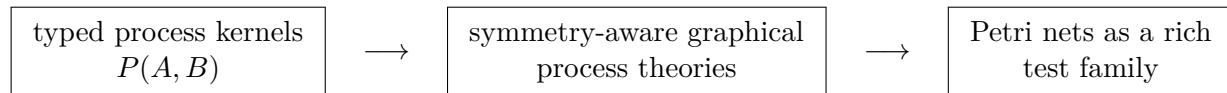
- Read  $\mathcal{D}_S = [\mathbb{B}_S^{op} \times \mathbb{B}_S, \mathbf{Set}]$  both as a category of typed process kernels and as the fixed-palette bicollection category underlying  $\Sigma$ -nets; distinguish net generators from their free execution theories.
- Exploit profunctor composition and Day convolution as series and parallel process composition.
- Distinguish kernel/endofunctor presentation from  $\boxtimes$ -locality: inhibitor and reset behaviour may be global kernels without being generated by local transitions in parallel with idle context.
- Study measuring objects between process theories and calculate concrete runners for implementation, regulation, observation and quantitative execution.
- Organise the family over changing palettes and compare it with open-system and cubical/HDA constructions.

### Open questions

- Classify or otherwise obtain usable descriptions of  $\odot$ -comonoids, i.e. procomonads, on  $\mathbb{B}_S$ , and compute the resulting measuring objects.
- What weakened one-sided measuring notion, if any, gives a nontrivial prop-level runner without forcing the residual to be a  $\boxtimes$ -comonoid?
- Can one write down useful  $\{T, R\}$ -coalgebras for the concrete examples above rather than merely assert that they should exist?
- Which Petri variants are best handled by changing the value category, which by changing the boundary/marking base, and which require leaving cocontinuous endofunctors altogether?
- How do measuring objects behave under base change along the ordered/symmetric/commutative maps  $S^* \rightarrow \mathbb{B}_S \rightarrow \mathbb{N}[S]$ ?
- How does the double category of open systems compare with the double category  $\mathbf{Run}(\mathcal{D}_S)$ ?
- Can the known Petri-to-HDA semantics be promoted to a sufficiently monoidal comparison to induce a map between Run worlds?

## 16 The larger lesson

The chapter begins with the observation that many systems are naturally typed by two resource boundaries and admit both sequential and parallel composition. That leads to the progression



Petri nets are valuable precisely because they stress almost every part of the architecture. Whole-grain and  $\Sigma$ -nets force us to treat symmetry seriously. Open nets force us to distinguish process boundaries from external system interfaces. Coloured and guarded nets force us to distinguish place types from token data. Inhibitor and reset arcs expose the limit of a fixed local  $A \rightarrow B$  description. Timed and stochastic nets test enrichment and residual target theories. The HDA translation reconnects graph-shaped process semantics with the cubical geometry of Part VII.

**Part VIII in one sentence.** The category  $\mathcal{D}_S$  is best read as a category of typed process kernels, simultaneously presheaf-like and endofunctor-like; Petri nets then form a particularly revealing family of process theories inside and around this setting, rich enough to test symmetry, wiring, quantitative extension, state-dependent enabling, and comparison with higher-dimensional concurrency.

## References

- [1] A. Joyal. Foncteurs analytiques et espèces de structures. In *Combinatoire énumérative*, Lecture Notes in Mathematics 1234, Springer, 1986.
- [2] N. Gambino and A. Joyal. On operads, bimodules and analytic functors. arXiv:1405.7270, 2014.
- [3] J. Kock. Graphs, hypergraphs, and properads. *Collectanea Mathematica* 67:155–190, 2016. arXiv:1407.3744.
- [4] J. Kock. Whole-grain Petri nets and processes. *Journal of the ACM* 70(1), 2023. arXiv:2005.05108.
- [5] J. C. Baez, F. Genovese, J. Master, and M. Shulman. Categories of Nets. Proceedings of LICS 2021. arXiv:2101.04238.
- [6] T. Booker and R. Street. Tannaka duality and convolution for duoidal categories. arXiv:1111.5659, 2011.
- [7] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1*. Springer, 2nd edition, 1996.
- [8] J. C. Baez and J. Master. Open Petri Nets. *Mathematical Structures in Computer Science* 30(3), 2020. arXiv:1808.05415.

- [9] J. C. Baez and K. Courser. Structured Cospans. *Theory and Applications of Categories* 35, 2020. arXiv:1911.04630.
- [10] J. Master. Petri nets based on Lawvere theories. *Mathematical Structures in Computer Science* 30(7):833–864, 2020. arXiv:1904.09091.
- [11] J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation* 88(2):105–155, 1990.
- [12] M. Hyland, I. López Franco, and C. Vasilakopoulou. Hopf measuring comonoids and enrichment. *Proceedings of the London Mathematical Society* 115(5):1118–1148, 2017. arXiv:1509.07632.
- [13] A. Amrane, H. Bazille, U. Fahrenberg, L. Helouet, and P. Schlehuber-Caissier. Petri Nets and Higher-Dimensional Automata. arXiv:2502.02354, 2025.