From algebraic effects to Sweedler runners

16 October 2026

The concept of a *runner* emerged in computer science from attempts to understand how computations described by algebraic operations are executed by stateful machines. Its development brought together several initially distinct strands of research: the algebraic theory of computational effects, the coalgebraic study of comodels, programming languages with handlers and resources, and the categorical theory of interactions between monads and comonads. These strands met in the Sweedler theory of monads developed by McDermott, Rivas and Uustalu (2022), itself an application of a longer algebraic tradition of measuring comonoids.

Our approach builds on this convergence. We regard a runner not merely as a means of executing a program, but as a general description of how one system realises the operations of another, possibly using its own residual effects. This interpretation opens the possibility of applying Sweedler theory to dynamical systems, machines, verification, and compositional system design.

1 Algebraic effects and comodels: the prehistory

The mathematical starting point is the use of monads to describe computational effects, established by Moggi (1989, 1991) and subsequently developed through the algebraic theory of effects by Plotkin and Power. The central idea is that a computation can be described in terms of operations and equations, without committing to how those operations are implemented. A theory might provide operations for reading and writing memory, issuing commands, making choices, or observing an environment. Its equations specify which combinations of operations are regarded as equivalent.

A monad T packages such a computational theory. Its algebras provide interpretations of computations; algebraic effect handlers use such interpretations to give meanings to operations, possibly passing on residual effects. There is also a dual notion, that of a *comodel*. A comodel responds to operations by producing observations and updating its own state. This coalgebraic perspective was developed by Power and Shkaravska in *From Comodels to Coalgebras: State and Arrays* (2004). They showed how comodels of Lawvere theories give rise to comonads and investigated their relationship to state and arrays. Plotkin and Power subsequently developed the interaction between models and comodels in operational semantics (2008).

The distinction between models and comodels is fundamental. An algebra interprets operations in a space of values; a comodel describes an environment that performs operations by observing and changing its state. Thus, while algebras and handlers interpret or eliminate operations, comodels and runners supply stateful environments that execute them. This algebra/handler versus comodel/runner distinction becomes especially informative when the execution leaves a residual effect theory. For systems theory, this suggests interpreting an algebraic theory as a specification of possible operations and a comodel as a machine that implements them.

2 Møgelberg and Staton (2011–2014): linear usage of state

An important precursor to the programming-language treatment of runners is the work of Rasmus Ejlers Møgelberg and Sam Staton, presented at CALCO 2011 and developed in *Linear Usage of State* (2014). They investigate the sense in which every monad can be presented as a *linear state monad*. Their fully complete state-passing translation takes an effectful call-by-value language into the enriched call-by-value calculus, where a state resource is used linearly. The analysis

is not confined to mutable store: it encompasses general algebraic effects, even when they are not ordinarily regarded as stateful. The corresponding categorical semantics uses enriched adjunctions.

The qualification *linear* is essential. State is threaded through execution rather than being freely copied, discarded, or replaced; this makes the state-passing representation much more general than ordinary programming with a freely manipulable store. The authors also relate generic effects to state-access operations, so that the relationship between algebraic operations and their stateful interpretation is part of the construction.

This is a direct antecedent of Ahman and Bauer's runners. As they explicitly observe in *Runners in Action*, Møgelberg and Staton's linear-use state-passing translation relies on equipping state with a comodel structure for the effects being executed; their own runner semantics arises when that setup is specialised to a particular Kleisli adjunction. The linear-state tradition thus supplies a distinct but closely connected strand alongside Uustalu's identification of stateful runners with comodels. For our later systems interpretation, it also highlights the difference between algebraic equations constraining an implementation and resource discipline governing how its state may be used. The abstract Sweedler coalgebra correspondence alone should not be taken to establish all the linearity and finalisation guarantees of a programming calculus.

3 Uustalu (2015): stateful runners

The identification of comodels with *stateful runners* was made explicit by Tarmo Uustalu in *Stateful Runners of Effectful Computations* (2015). Uustalu asked what structure a set of states S must possess in order to execute computations described by a given monad T. His answer was a monad morphism

$$T \longrightarrow \mathrm{St}_S, \qquad \mathrm{St}_S(X) = (S \times X)^S.$$

Such a morphism interprets each computation as a state-transforming computation: given an initial state, it produces a result together with a new state.

Uustalu proved that, when T corresponds to a Lawvere theory, equipping S with such a runner is equivalent to equipping it with a comodel of that theory. This result is conceptually decisive. It identifies the abstract notion of a comodel with the concrete idea of a machine that executes effectful operations.

Examples illustrate the range of the construction. A runner for interactive input and output must respond to requests and process commands, and a runner for global state carries the structure of a (very well-behaved) lens. Equations can also forbid execution altogether: for binary choice subject to the semilattice laws, commutativity forces any comodel to be empty, whereas free binary choice admits choice-resolving runners. Indeed, a comodel operation $S \to S + S \cong 2 \times S$ that obeys commutativity would have to give the same branch on either ordering of arbitrary continuations, which is impossible when S is inhabited. Thus the finite-powerset theory has no nonempty ordinary stateful runner. In each case, the operations belong to the computational theory, while their implementation belongs to the runner.

There is, however, an important restriction in this initial formulation. Execution is interpreted into an ordinary state monad. Once an operation has been executed, there are no additional computational effects beyond state transformation. A natural generalisation is to allow the implementing machine to have effects of its own.

4 Ahman and Bauer (2020): runners as programming constructs

The programming-language significance of runners was developed by Danel Ahman and Andrej Bauer in *Runners in Action* (ESOP 2020). Their work takes runners beyond an abstract semantic definition and makes them into constructs for organising effectful programs. The associated

calculus, λ_{coop}, describes how computations interact with resources and how runners can be composed and nested.

Building on Møgelberg and Staton, Ahman and Bauer use a linear-state discipline to control the use of resources by computations executing under a runner. They also explicitly identify their construction as a specialisation of the earlier enriched state-passing semantics to a Kleisli adjunction. Resource management is particularly prominent. A runner may provide access to an external resource, such as a file or execution environment, while controlling how the resource is used and ensuring that appropriate finalisation takes place. The examples include virtual machines, sandboxing, cost instrumentation, and stateful resources. The paper also gives a formal treatment of resource usage and finalisation, together with implementations demonstrating that runners can serve as practical programming abstractions.

An important conceptual point is that the computational environment itself can be layered. A program may execute inside a sandbox, which executes inside a virtual machine, which in turn executes inside a larger computational environment. This makes runners relevant to compositional systems design. A component need not implement all its operations directly. It may delegate some of them to another component, while supplying its own state and enforcing additional constraints.

The calculus also accounts for exceptions, kill signals and finalisation. The bare monad-morphism semantics of a runner captures a mathematical core, not automatically the full operational discipline of λ_{coop}; exceptions may be modelled in a suitable residual monad, while signals and finalisation require explicit treatment. Although the primary motivation is programming-language semantics, the same idea applies to interacting machines. A system can realise the interface of another system while mediating its access to an underlying environment.

5 Katsumata, Rivas and Uustalu (2020): interaction laws and Sweedler duality

A different but complementary development was made by Shin-ya Katsumata, Exequiel Rivas and Tarmo Uustalu in *Interaction Laws of Monads and Comonads* (LICS 2020). The paper investigates mathematical laws describing interactions between effectful computations and the behaviours of machines that execute effects. Here monads represent computational operations, while comonads represent the possible behaviours of their environments. An interaction law specifies how these two kinds of structure communicate, coherently with their respective monad and comonad structures.

The authors show that monad–comonad interaction laws can be understood using suitable generalisations of duality and Sweedler duality. In particular, under appropriate hypotheses, a monad T possesses a universal interacting comonad, its *Sweedler dual* (sometimes denoted $T^\bullet$). Its coalgebras can be understood as stateful runners for T, equivalently as comodels in the setting of the preceding results. We reserve the relative Sweedler-hom notation $\mathcal{M}(T, R)$ for the subsequent development by McDermott, Rivas and Uustalu. The universal construction is significant. Rather than separately defining the coalgebraic structure appropriate to every computational theory, it constructs a canonical comonad from the theory itself.

Katsumata, Rivas and Uustalu also place interaction laws within a broader categorical setting involving Day convolution, Chu constructions and monoidal structures. This work therefore reveals that the relationship between computations and their executing machines is not peculiar to state monads. It belongs to a general theory of interactions between algebraic and coalgebraic structures. The next step is to allow the interaction to leave a nontrivial computational residue.

6 The algebraic ancestry of Sweedler measuring

The name *Sweedler* points to a mathematical lineage older than runners. In the theory of algebras and coalgebras, Sweedler's work on Hopf algebras (1969) introduced measuring coalgebras: a

coalgebra can parametrise maps between algebras in a way compatible with multiplication and units. The universal measuring coalgebra $P(A, B)$ packages all such measurements, rather than choosing any particular one. The finite or Sweedler dual is a related classical construction.

Later categorical developments include Porst and Street's study of generalised Sweedler duals, Anel and Joyal's treatment of Sweedler theory, and Hyland, López Franco and Vasilakopoulou's theory of universal measuring comonoids and enrichment. López Franco and Vasilakopoulou then extend measuring comonoids to *duoidal categories*, which carry two compatible tensor products. This is the abstract setting in which the relative Sweedler construction used for monads becomes available.

The point of the 2022 work is therefore not the invention of measuring comonoids or of duoidal Sweedler theory. It is their instantiation in the duoidal category of endofunctors, with composition and Day convolution, and the recognition that the resulting measurements encode interactions and residual stateful runners.

7 McDermott, Rivas and Uustalu (2022): the Sweedler theory of monads

The decisive application to computational monads was provided by Dylan McDermott, Exequiel Rivas and Tarmo Uustalu in *Sweedler Theory of Monads* (FoSSaCS 2022). Their work interprets monad–comonad interaction laws as measuring maps in the Sweedler theory of duoidal categories. This makes available not only the Sweedler dual of a single monad, but a more general construction involving two monads T and R.

Its central object is the *Sweedler hom* (using the notation of the accompanying tutorial)

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

a comonad describing interactions in which computations governed by T may be executed using residual computations governed by R. A coalgebra

$$\xi : S \longrightarrow \mathcal{M}(T, R)(S)$$

may consequently be interpreted as a stateful runner from T to R.

For a locally presentable base and accessible monads T, R, the relevant Sweedler construction exists under the standard hypotheses; in the set-based state-passing case, its coalgebras on S correspond to monad morphisms

$$T \longrightarrow \mathrm{St}_S(R), \qquad \mathrm{St}_S(R)(X) = \big(R(S \times X)\big)^S.$$

Thus, a computation described by T, when supplied with a state in S, can produce a computation described by R, whose eventual result includes a new state. Taking $R = \mathrm{Id}$ recovers the ordinary stateful runners of Uustalu. Allowing R to be nontrivial captures implementations that delegate operations, invoke other effects, or interact with a surrounding computational environment.

McDermott, Rivas and Uustalu also develop the corresponding *Sweedler copower*, together with algebraic and coalgebraic characterisations of the universal constructions. The algebraic/coalgebraic analysis of interaction laws in Uustalu and Voorneveld (APLAS 2020) is another immediate predecessor and helps explain how the universal Sweedler picture connects with handling and execution. The importance of this contribution is structural. Runners with residual effects are no longer isolated semantic devices: they arise from universal constructions in a general categorical framework.

For our purposes, this paper supplies the principal mathematical foundation. The measuring comonad $\mathcal{M}(T, R)$, its universal property, and its coalgebraic interpretation are established ingredients of the theory, not innovations introduced by our programme.

8 Garner (2021–2022): the category of behaviours

A further development, especially important for connecting runners to systems theory, is Richard Garner's *The Costructure–Cosemantics Adjunction for Comodels for Computational Effects*, published online in 2021 and in journal form in 2022. Garner investigates the passage from an algebraic theory to the comonad whose coalgebras are its comodels. He shows that this belongs to an adjunction between suitable categories of monads and comonads.

A particularly striking result is that the comonad associated with an accessible computational theory can be represented through a small *category of behaviours*. Writing this category as $\mathcal{B}_T$, one obtains an equivalence, with the appropriate variance convention,

$$\mathrm{Comod}(T) \simeq [\mathcal{B}_T, \mathbf{Set}].$$

The objects and morphisms of $\mathcal{B}_T$ encode respectively the static and dynamic aspects of the behaviours available to comodels. Consequently, the category of machines implementing a theory has a presheaf-like representation. This brings it into contact with a substantial body of categorical machinery, including presheaf toposes, universal constructions, and methods for reasoning about behavioural equivalence.

For accessible T over **Set**, Garner's comodel comonad can be identified, by their shared universal characterisation, with the absolute interacting comonad (the Sweedler dual), namely $\mathcal{M}(T, \mathrm{Id})$ in MRU's relative notation. This identification uses the compatible hypotheses and conventions; it is not a claim about arbitrary monads in arbitrary bases. Thus Garner and Katsumata–Rivas–Uustalu describe the same absolute object from different starting points, while MRU also describes its residual generalisation. The polynomial nature of the comodel comonad is crucial here: Ahman and Uustalu's *Directed Containers as Categories* (2016) identifies polynomial comonads with small categories at the level of objects, and explains their nonstandard morphism variance. Their coalgebras are copresheaf-like actions of these categories, which accounts for the displayed equivalence. The combination suggests studying not only the behaviours of individual theories but also how those behaviours transform under runners with nontrivial residue.

9 Libkind and Spivak (2024–2025): pattern runs on matter

A particularly direct bridge from the mathematics of computation to a general theory of systems is Sophie Libkind and David Spivak's *Pattern Runs on Matter: The Free Monad Monad as a Module over the Cofree Comonad Comonad*. First circulated in 2024 and published in the ACT 2024 proceedings in 2025, this paper was a significant motivation for the present programme. Its guiding distinction is between a *pattern*, which specifies how an interaction may proceed, and the *matter* on which that pattern runs, which supplies responses as the interaction unfolds. Their examples are deliberately not confined to computer programs: interviews run on people, voting schemes on voters, and games on players, as well as programs on operating systems.

The distinction is formalised in the category **Poly** of polynomial functors. For a polynomial p, the free monad $\mathfrak{m}_p$ describes well-founded decision trees generated by p: a pattern can pose questions, branch on answers, and eventually terminate. For a polynomial q, the cofree comonad $\mathfrak{c}_q$ describes potentially ongoing behavioural trees: matter can continue to supply responses. The authors establish a natural module action

$$\Xi_{p,q} : \mathfrak{m}_p \otimes \mathfrak{c}_q \longrightarrow \mathfrak{m}_{p \otimes q},$$

where $\otimes$ denotes the Dirichlet tensor product of polynomials. This action gives a precise mathematical meaning to a pattern *running on* matter. For example, an input list may be run against the continuing behaviour of a Moore machine to produce an output list.

This work makes explicit a philosophical distinction that is central to our approach: the specification of an interaction should not be conflated with the system that carries it out. It

also gives a compelling reason to seek categorical structure for the pairing itself, rather than to study either side in isolation. It connects directly with Spivak's broader polynomial approach to dynamical systems, in which interfaces, dependent lenses, and composition are primary tools, and with the coalgebraic treatment of continuing behaviours.

There is nevertheless an important mathematical distinction. The module action Ξ concerns the *free-monad/cofree-comonad constructions on polynomial signatures*; it is not, simply by a change of notation, MRU's relative Sweedler hom $\mathcal{M}(T, R)$. The former explains how a terminating pattern interacts with an ongoing substrate. The latter universally classifies executions of operations satisfying a specified monad's equations, potentially with a nontrivial residual monad. Relating these constructions precisely—particularly when a free theory is quotiented by laws, and when residue is present—is part of the proposed research, not an identity already proved by either paper.

The connection also helps sharpen the role of laws in the systems interpretation. An arbitrary decision-tree pattern need not satisfy any equations beyond those of its free monad, whereas a quotient theory can insist that its implementations obey chosen equations. Such equations may restrict, or even eliminate, comodels. The Sweedler viewpoint thus offers a way to ask not only *what pattern runs on what matter?*, but *which material realisations genuinely implement a specified operational theory, and what residual interactions and guarantees do they preserve?*

10 Niu–Spivak and Myers: neighbouring theories of dynamical systems

The transition from computational runners to general systems must be compared with existing categorical theories of dynamics. Two especially important points of comparison are Niu and Spivak's *Polynomial Functors: A Mathematical Theory of Interaction* (2025) and David Jaz Myers' *Categorical Systems Theory* (2023 manuscript). Both already offer general, compositional ways to understand systems; our claim cannot be that dynamical systems, interfaces, or their wiring first acquire categorical descriptions through Sweedler theory.

10.1 Niu and Spivak: interfaces and dependent lenses

Niu and Spivak present polynomial functors as a language for interacting systems. Their Chapter 4, *Dynamical Systems as Dependent Lenses*, treats Moore machines and discrete-time dynamical systems through lenses between polynomial interfaces. It explains how systems are transformed and interconnected, including parallel and composite constructions. Chapter 7 develops polynomial comonoids and retrofunctors and, in particular, the relationship between polynomial comonoids and categories. This is more than a passing analogy with our work: the objects, maps, and diagrams used to organise stateful interaction inhabit the same polynomial world in which the Sweedler constructions can sometimes be investigated.

There are nevertheless different starting points. A lens describes a protocol or interaction across an interface and is naturally adapted to wiring. A runner for an operational monad T also specifies how *equations between operations* are respected by an executing machine. Those equations can rule out possible implementations even when the corresponding unequated polynomial signature has abundant coalgebras. Thus there is a potentially useful separation of *signature/interface*, *quotient operational theory*, and *law-abiding implementation*. This does not establish that lenses cannot express the same restrictions with additional structure; it identifies an issue that the runner presentation puts at the centre.

The comparison should be mathematical rather than terminological. For a concrete Moore theory, we should relate its runner/comodel category to Niu–Spivak's lens presentation of Moore machines, account for composition and wiring on both sides, and determine exactly what is changed when the target residue R is nontrivial. The equivalence of polynomial comonoids with categories, together with Garner's category of behaviours, supplies a further bridge to the

book's second part. A general comparison theorem, however, has not yet been established by our programme.

10.2 Myers: dynamics, lenses, charts and system morphisms

Myers' *Categorical Systems Theory* provides a broad framework in which dynamical systems are organised through interfaces, lenses, charts, and categorical constructions adapted to their state spaces. It is not restricted to discrete-time automata: smooth dynamics, for example, can be represented by vector fields $v : S \to TS$, with morphisms and trajectories formulated in a systematic categorical language. The use of lenses and charts also makes distinctions between kinds of system comparison and transformation. This is an important predecessor for any claim to a general categorical theory of dynamical systems.

Our prospective double category $\mathrm{Run}(\mathcal{D})$ introduces a different distinction: *tight* theory interpretations translate specified operations, while *loose* stateful runners implement them, perhaps with residual effects. Whether particular systems-and-charts constructions admit a comparison functor to a suitable runner double category, or vice versa, is an open question. The two perspectives may be complementary rather than one subsuming the other: geometric notions of trajectory and comparison are prominent in Myers' setting, while equations of operations and their residual execution are prominent in ours. Comparisons with DOTS and related interface-based approaches belong here as well, but should not be conflated with a demonstrated equivalence of frameworks.

10.3 A discriminating test

The most useful first experiment is to work through *one and the same Moore machine* in all three settings. We should ask: (i) whether the ordinary polynomial-lens or charts description is recovered from the relevant Sweedler comodels or runners; (ii) whether nontrivial residue gives a canonical description of delegation that requires extra choices on the lens side; and (iii) whether quotienting the operational theory by equations leads to a useful compositional safety or behavioural theorem. The third question is especially important: a new notation for the same transition systems is less compelling than a theorem explaining how admissibility or a system guarantee survives implementation and wiring.

Libkind and Spivak's pattern–matter construction is a further motivation for this comparison. It makes the pairing between a finite, terminating interaction and an ongoing substrate explicit; the question for relative Sweedler theory is how this pairing behaves in the presence of specified equations, residual effects, and verification properties. These are research questions, not claims of priority for the existing theories of dynamical systems.

11 From runners of programs to a theory of systems

These developments provide a substantial foundation for a broader categorical systems theory. The basic proposal is to regard an algebraic theory T as specifying the operations available at a system interface, including the equations those operations must satisfy. A runner then describes a realisation of that interface by a state-bearing machine, potentially using a different collection of operations R. This interpretation applies to familiar computational examples, but it also extends naturally to dynamical systems, automata, physical models, and networks of interacting components.

The distinction between a theory and its implementations is particularly useful when equations express physical or engineering constraints. Rather than beginning with arbitrary transition systems and subsequently selecting those that satisfy the required laws, one may incorporate the laws into the operational theory itself. Its comodels are then law-abiding implementations by construction.

The residue R adds another dimension. An implementing system need not be self-contained: it may call upon external services, delegate commands, use probabilistic effects, or interact with a surrounding environment. The distinction between the operations exposed by a component and those used to realise them becomes mathematically explicit.

This perspective sits alongside the coalgebraic tradition of Rutten and the polynomial and categorical systems frameworks compared in the preceding section. The proposed contribution is not another claim to have introduced compositional interfaces or dynamical systems into category theory. Rather, it is a systematic account of law-governed operational theories, stateful realisations, residual execution, and the possible transfer of behavioural and verification properties. The precise comparison with established lens, chart, and DOTS methods is an explicit part of the research agenda.

Our research seeks to develop these ideas in several directions.

- First, we study how runners compose and how components can be wired together, including situations in which operations of one theory are implemented by composite computations of another.
- Second, we study the behavioural semantics of runners, extending the perspective of Garner's behaviour categories towards relative behaviour, observability, and minimal realisation.
- Third, we vary the ambient category. In addition to sets, we consider categories carrying predicates, costs, spatial information, probabilistic structure, concurrency, or smooth dynamics. The intention is to make the same runner-based constructions available in settings that encode properties relevant to systems engineering.
- Fourth, we investigate how safety properties and other specifications can be transported through runners and preserved under system composition. This is especially important for applications in which the objective is not merely to describe a system, but to establish guarantees about its operation.

These are research directions rather than consequences automatically guaranteed by the existing Sweedler theory. Their development requires additional constructions and, in some cases, new existence or compatibility results for the chosen ambient categories.

12 Perspective

The history of runners exhibits a gradual broadening of scope. The theory begins with the distinction between algebraic operations and the environments that execute them. Møgelberg and Staton establish a linear-use state-passing semantics for general computational effects; Uustalu identifies these executing environments with comodels. Ahman and Bauer show how runners can organise real programs and manage resources. Katsumata, Rivas and Uustalu connect execution to interaction laws and Sweedler duality. McDermott, Rivas and Uustalu establish the general Sweedler theory of monads, including residual effects. Garner provides a complementary understanding of the behavioural categories associated with computational theories.

Libkind and Spivak, meanwhile, articulate the pattern–matter pairing that gives the passage from computational semantics to systems theory a particularly direct motivation. Our systems-theoretic programme begins where these developments meet. In particular, its planned temporal and graded extensions build on this general history rather than being presupposed by it. Its guiding question is no longer only *how can a computation be run?* but also:

How can one system realise another system's operations, what behaviour results from that realisation, and which properties survive the interaction and composition of systems?

The promise of Sweedler theory is that these questions can be posed within a common categorical framework, while retaining the operational meaning that originally motivated runners in computer science.

13 References

1. Moggi, E. (1989). *Computational Lambda-Calculus and Monads.* LICS 1989, 14–23.
2. Moggi, E. (1991). *Notions of Computation and Monads.* Information and Computation 93(1), 55–92.
3. Plotkin, G., and Power, J. (2002). *Notions of Computation Determine Monads.* FoSSaCS 2002, LNCS 2303, 342–356.
4. Plotkin, G., and Power, J. (2003). *Algebraic Operations and Generic Effects.* Applied Categorical Structures 11, 69–94.
5. Power, J., and Shkaravska, O. (2004). *From Comodels to Coalgebras: State and Arrays.* Electronic Notes in Theoretical Computer Science 106, 297–314.
6. Plotkin, G., and Power, J. (2008). *Tensors of Comodels and Models for Operational Semantics.* Electronic Notes in Theoretical Computer Science 218, 295–311.
7. Sweedler, M. E. (1969). *Hopf Algebras.* W. A. Benjamin.
8. Porst, H.-E., and Street, R. (2016). *Generalizations of the Sweedler Dual.* Applied Categorical Structures 24, 619–647.
9. Anel, M., and Joyal, A. *Sweedler Theory.* Research manuscript; bibliographic details to be confirmed.
10. Hyland, M., López Franco, I., and Vasilakopoulou, C. (2017). *Hopf Measuring Comonoids and Enrichment.* Proceedings of the London Mathematical Society 115(5), 1118–1148.
11. López Franco, I., and Vasilakopoulou, C. *Duoidal Categories, Measuring Comonoids and Enrichment.* arXiv:2005.01340 (2020 preprint).
12. Møgelberg, R. E., and Staton, S. (2014). *Linear Usage of State.* Logical Methods in Computer Science 10(1:17), 1–52. https://doi.org/10.2168/LMCS-10(1:17)2014. (Expanded version of work presented at CALCO 2011.)
13. Uustalu, T. (2015). *Stateful Runners of Effectful Computations.* Electronic Notes in Theoretical Computer Science 319, 403–421.
14. Ahman, D., and Uustalu, T. (2016). *Directed Containers as Categories.* Electronic Proceedings in Theoretical Computer Science 207, 89–98.
15. Ahman, D., and Bauer, A. (2020). *Runners in Action.* ESOP 2020, LNCS 12075.
16. Katsumata, S., Rivas, E., and Uustalu, T. (2020). *Interaction Laws of Monads and Comonads.* LICS 2020, 604–618.
17. Uustalu, T., and Voorneveld, N. (2020). *Algebraic and Coalgebraic Perspectives on Interaction Laws.* APLAS 2020, LNCS 12470, 186–205.
18. Garner, R. (2022). *The Costructure–Cosemantics Adjunction for Comodels for Computational Effects.* Mathematical Structures in Computer Science 32(4), 374–419.
19. Libkind, S., and Spivak, D. I. (2025). *Pattern Runs on Matter: The Free Monad Monad as a Module over the Cofree Comonad Comonad.* Proceedings ACT 2024, EPTCS 429, 1–28. https://arxiv.org/abs/2404.16321.

20. Niu, N., and Spivak, D. I. (2025). *Polynomial Functors: A Mathematical Theory of Interaction.* London Mathematical Society Lecture Note Series 498, Cambridge University Press. In particular Chapters 4 and 7. https://doi.org/10.1017/9781009576734.

21. Myers, D. J. (2023). *Categorical Systems Theory.* Manuscript (version dated September 3, 2023). https://www.davidjaz.com/Papers/DynamicalBook.pdf.

22. Rutten, J. J. M. M. (2000). *Universal Coalgebra: A Theory of Systems.* Theoretical Computer Science 249(1), 3–80.

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