

Getting to Grips with Sweedler Runners for Systems

A conversational tutorial, Part I: from the identity monad to Moore machines

September 2026

Aim. We want to understand the Sweedler hom $\mathcal{M}(T, R)$ through its coalgebras: R -residual stateful runners for T . Rather than begin abstractly, we work through a sequence of tiny examples until an ordinary Moore machine appears as a runner. A runner S induces a handler functor whose underlying carrier is an exponential A^S , not a product $A \times S$. The product reappears on the other side of the adjunction, in the model-comodel tensor.

Contents

1	The basic picture	2
1.1	The handler functor, and the adjunction we will use	2
2	First case: $T = R = \text{Id}$	3
3	One unary operation: $T = \mathbb{N}y$, $R = \text{Id}$	3
3.1	The runner-induced handler	4
3.2	The tensor/execution side	4
3.3	A clock	4
4	Unary dynamics on both sides: $T = R = \mathbb{N}y$	5
4.1	The induced handler is still an exponential	5
5	Branching: $T = F(y^O)$, $R = \text{Id}$	6
5.1	Running a whole tree	6
5.2	The induced handler	7
5.3	The tensor relation	7
6	Quotienting to the reader monad y^O	7
6.1	The induced reader handler	8
7	Assembling the Moore machine	8
7.1	The Moore machine acting on handlers	9
7.2	Executing a handler against the Moore machine	9
7.3	The Moore machine as an interpreter	10
8	Where we have got to	10
9	What comes next: residual effects	10

References and orientation

11

1 The basic picture

Let T and R be monads on **Set**. An R -residual stateful runner for T , with carrier S , is a family of maps

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

natural in X , satisfying the unit and multiplication compatibility laws. Equivalently, it is a monad morphism

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

The Sweedler hom $\mathcal{M}(T, R)$, when it exists, is the comonad whose coalgebras are precisely these runners.

The systems reading is already visible in the formula. A T -computation, together with the current runner state $s \in S$, is re-expressed as an R -computation whose result contains both an ordinary return value and a new runner state. Thus a runner is a stateful mechanism for realising T -behaviour using R -behaviour.

For this part of the tutorial we mostly take $R = \mathrm{Id}$. Then

$$\theta_X : TX \times S \longrightarrow X \times S,$$

and the runner completely discharges the T -effect. In the Lawvere-theory setting these are exactly *comodels*.

1.1 The handler functor, and the adjunction we will use

A runner S also induces a functor on Eilenberg–Moore algebras

$$\Theta_S : \mathbf{Alg}(R) \longrightarrow \mathbf{Alg}(T).$$

The underlying carrier functor is

$$A \longmapsto A^S.$$

This is the statement proved abstractly in the runner/interaction-law account: the runner lifts the internal-hom functor $S \Rightarrow -$ from the base category to the two algebra categories.

In our **Set**-based examples, this functor has a left adjoint. When $R = \mathrm{Id}$, that left adjoint is the familiar model–comodel tensor:

$$- \otimes_T S \dashv \Theta_S,$$

so that

$$\mathbf{Set}(B \otimes_T S, X) \cong \mathbf{Alg}(T)(B, X^S).$$

It is useful to read the two sides as follows:

- X^S : use the runner to turn an ordinary set X into a T -handler;
- $B \otimes_T S$: execute a T -handler B against the runner S , quotienting by the equations that identify abstract operations with what the runner actually does.

There is generally no further adjoint on the right of Θ_S . Already on underlying sets, $(-)^S$ usually fails to preserve coproducts when $|S| > 1$, so it is generally not a left adjoint.

Terminology warning. In different parts of the literature, and in discussions of loose morphisms, words such as “induction” and “delegation” can be assigned to different directions of related adjunctions. To avoid muddle, in this tutorial we use the neutral phrase *runner-induced handler functor* for $\Theta_S : \mathbf{Alg}(R) \rightarrow \mathbf{Alg}(T)$, and call its left adjoint the *tensor/execution functor*.

2 First case: $T = R = \text{Id}$

Start with the most degenerate possible case. The Sweedler hom is again the identity comonad:

$$\mathcal{M}(\text{Id}, \text{Id}) \cong \text{Id}.$$

A coalgebra for the identity comonad is a set S with a map $S \rightarrow S$ satisfying the counit law. That law forces the map to be the identity. So a runner contains no genuine dynamics:

$$\text{an } (\text{Id}, \text{Id})\text{-runner is simply a set } S.$$

Similarly, an algebra for the identity monad is just a set. So if X is an R -handler, the runner-induced T -handler is simply

$$\Theta_S(X) = X^S.$$

The left adjoint is the ordinary product:

$$S \times - \dashv (-)^S.$$

Thus our very first handler adjunction is

$$\mathbf{Set}(S \times B, X) \cong \mathbf{Set}(B, X^S).$$

Nothing specifically “Sweedler” is happening yet; this is merely the ordinary product–exponential adjunction seen as the zero-th case of the story.

A useful correction to keep in mind from the beginning: a *set with an arbitrary endofunction* is not an algebra or coalgebra for Id . The identity laws force the endomap to be the identity. Arbitrary endofunctions appear one step later, from the free monad on one unary operation.

3 One unary operation: $T = \mathbb{N}y$, $R = \text{Id}$

Let

$$T(X) = \mathbb{N} \times X.$$

This is the free monad on one unary operation, equivalently the writer monad for the additive monoid $\mathbb{N}$. A T -algebra is exactly a set B equipped with an endofunction

$$g : B \rightarrow B.$$

The algebra map $\mathbb{N} \times B \rightarrow B$ sends (n, b) to $g^n(b)$.

From naturality considerations, a $\{T, \text{Id}\}$ -runner is the same as a T -comodel. It is a set S with an endofunction

$$f : S \rightarrow S.$$

This is our first genuine runner state. One abstract unary operation advances the runner by one step.

3.1 The runner-induced handler

Let X be an $R = \text{Id}$ -handler, hence just a set. The induced T -handler has carrier

$$X^S.$$

Its unary operation is

$$h \longmapsto h \circ f,$$

or pointwise

$$(\text{step}h)(s) = h(f(s)).$$

Thus the runner's forward dynamics on states induces a backwards pullback action on state-indexed continuations.

This contravariance is familiar in several guises. If $X = 2$, elements of X^S are predicates and the induced operation is a predicate transformer. If $X = \mathbb{R}$, they may be cost or value functions, pulled back along the transition.

3.2 The tensor/execution side

Now take a T -algebra (B, g) . The tensor $B \otimes_T S$ is obtained from $B \times S$ by imposing

$$(g(b), s) \sim (b, f(s)).$$

In words: performing the abstract unary operation in the handler is identified with leaving the handler state alone and advancing the runner.

The adjunction is

$$\mathbf{Set}(B \otimes_T S, X) \cong \mathbf{Alg}(T)(B, X^S).$$

This is the first place where the operational meaning of the tensor is easy to see. The relation literally lets an operation “flow across” from the model to the comodel.

3.3 A clock

Take

$$S = \mathbb{N}, \quad f(n) = n + 1.$$

The runner is a discrete clock. The induced operation on $X^{\mathbb{N}}$ is the shift

$$(\text{step}h)(n) = h(n + 1).$$

Thus the simplest nontrivial runner already turns an abstract unary action into temporal progression.

4 Unary dynamics on both sides: $T = R = \mathbb{N}y$

Now let both the abstract theory and the residual theory have one unary operation. An R -handler is a set A with endofunction

$$r : A \rightarrow A.$$

Because $T = \mathbb{N}y$ is the free monad on one unary operation, a runner

$$T \rightsquigarrow R$$

with residual state S is determined uniquely by the interpretation of that generator. Since

$$R(S) = \mathbb{N} \times S,$$

this is exactly a function

$$S \longrightarrow \mathbb{N} \times S,$$

or equivalently a pair of functions

$$f : S \rightarrow S, \quad k : S \rightarrow \mathbb{N}.$$

Thus these pairs do not describe merely a convenient family of runners: they describe all runners

$$\mathbb{N}y \rightsquigarrow \mathbb{N}y$$

with residual state S .

At state s , one abstract T -step is realised by $k(s)$ residual R -steps, after which the runner moves to $f(s)$. Iterating the abstract step therefore gives an accumulated number of residual steps

$$K_n(s) = \sum_{j=0}^{n-1} k(f^j(s)),$$

and after n abstract steps the runner state is $f^n(s)$. The cocycle-looking identity

$$K_{m+n}(s) = K_m(s) + K_n(f^m(s))$$

is exactly what one expects from composing runs.

4.1 The induced handler is still an exponential

The induced T -handler has carrier A^S . Its unary operation acts on $h : S \rightarrow A$ as follows:

$$\boxed{(\text{step}_T h)(s) = r^{k(s)}(h(f(s)))}.$$

The logic is simple: from state s , the runner says that one T -step requires $k(s)$ residual R -steps and updates its own state to $f(s)$. The continuation is therefore consulted at $f(s)$, and its result is acted on by $r^{k(s)}$.

The special case $k \equiv 1$ gives

$$(\text{step}_T h)(s) = r(h(f(s))).$$

So this is the first example in which the runner is not merely a clock: it is a small stateful translator from one dynamics to another.

It is tempting here to say that the induced handler has carrier $A \times S$ with transition $(a, s) \mapsto (r^{k(s)}a, f(s))$. That product system is certainly operationally meaningful, but it is *not* the canonical runner-induced Eilenberg–Moore handler functor of the Sweedler interaction law. The canonical functor has carrier A^S . Keeping these two constructions distinct prevents the adjunction from being accidentally reversed.

5 Branching: $T = F(y^O)$, $R = \text{Id}$

The unary examples are useful, but they only teach us about sequential dynamics. The next genuinely new phenomenon is branching.

Let $F(y^O)$ be the free monad on one O -ary operation. An element of $F(y^O)(X)$ is a finite O -branching tree whose leaves are labelled in X . A T -algebra is a set B with an arbitrary operation

$$\beta : B^O \rightarrow B.$$

There are no equations yet.

A comodel/runner consists of a cooperation

$$\gamma : S \rightarrow O \times S.$$

Write

$$\gamma(s) = (o(s), \sigma(s)).$$

The interpretation is concrete: when the runner encounters an O -way branching node, state s tells it to choose branch $o(s)$, and then the runner state changes to $\sigma(s)$.

5.1 Running a whole tree

The runner determines maps

$$\theta_X : F(y^O)(X) \times S \longrightarrow X \times S$$

recursively. At a leaf,

$$\theta_X(x, s) = (x, s).$$

At a branching node $(t_i)_{i \in O}$,

$$\theta_X((t_i)_{i \in O}, s) = \theta_X(t_{o(s)}, \sigma(s)).$$

So the runner walks through the tree: its state chooses a branch, changes, then chooses again at the next branch node, until a leaf is reached.

5.2 The induced handler

Given a plain set X , the induced T -handler has carrier X^S . Given an O -indexed family of continuations

$$(h_i : S \rightarrow X)_{i \in O},$$

the O -ary algebra operation is

$$\boxed{\alpha((h_i)_{i \in O})(s) = h_{o(s)}(\sigma(s)).}$$

The abstract operation supplies all O continuations. The runner decides which continuation is actually taken and what state it should be evaluated at.

5.3 The tensor relation

For a T -algebra (B, β) , the tensor $B \otimes_T S$ is the quotient of $B \times S$ generated by

$$\boxed{(\beta((b_i)_{i \in O}), s) \sim (b_{o(s)}, \sigma(s)).}$$

This is perhaps the clearest instance yet of “execution”: an abstract O -ary algebra operation is identified with the single branch which the runner actually selects.

6 Quotienting to the reader monad y^O

Now quotient the free branching theory by the equations of the reader monad

$$T(X) = X^O.$$

The algebraic operation **get** satisfies the familiar equations

$$\text{get}(i \mapsto x) = x$$

and

$$\text{get}(i \mapsto \text{get}(j \mapsto x_{ij})) = \text{get}(i \mapsto x_{ii}).$$

The first says that reading an environment and then ignoring what was read is the same as doing nothing. The second says that two successive reads are coherent: they see the same environment.

Start from a free comodel

$$s \mapsto (o(s), \sigma(s)).$$

Apply the first equation. On the left, the runner performs a **get** and therefore changes state to $\sigma(s)$, even though every branch returns the same value. On the right, no operation occurs, so the state stays s . Hence

$$\sigma(s) = s.$$

Thus the quotient immediately collapses the cooperation to

$$s \mapsto (o(s), s),$$

which is equivalent to nothing more than a map

$$\boxed{o : S \rightarrow O.}$$

The second reader equation is then automatic. Before imposing $\sigma = \text{Id}$, it would have forced

$$o(\sigma(s)) = o(s), \quad \sigma^2(s) = \sigma(s),$$

and the first equation strengthens this idempotence all the way to $\sigma = \text{Id}$.

The free O -ary theory gives a *stateful branch selector*

$$S \rightarrow O \times S.$$

The reader equations turn this into a *pure observation*

$$S \rightarrow O.$$

This is exactly the observation half of a Moore machine.

6.1 The induced reader handler

Given a set X , the runner $o : S \rightarrow O$ gives a reader algebra on X^S :

$$\text{get}((h_i)_{i \in O})(s) = h_{o(s)}(s).$$

Compared with the free case, the state update σ has disappeared. Reading chooses a branch but does not alter the state being read.

7 Assembling the Moore machine

We now combine the two pieces we have understood separately.

Take a theory with signature

$$Ay + y^O.$$

There is one unary operation step_a for each $a \in A$, together with one O -ary operation get . Impose the reader equations on get , and no equations relating the distinct unary operations to one another or to get . Call the resulting monad T_{Moore} .

A $\{T_{\text{Moore}}, \text{Id}\}$ -runner consists of:

- for each $a \in A$, a state transition $S \rightarrow S$, together giving

$$\delta : A \times S \rightarrow S;$$

- from the reader part, a pure observation

$$o : S \rightarrow O.$$

Therefore

$$\{T_{\text{Moore}}, \text{Id}\}\text{-runner} = (S, \delta, o),$$

which is precisely an ordinary Moore machine.

This is the point of the whole progression. We have not simply declared that a Moore machine is a runner. We have watched its transition and observation maps arise from two elementary pieces of effect syntax:

$$Ay \rightsquigarrow \delta : A \times S \rightarrow S,$$

and

$$y_{\text{reader}}^O \rightsquigarrow o : S \rightarrow O.$$

7.1 The Moore machine acting on handlers

Take an ordinary set X . The runner induces a T_{Moore} -handler on X^S .

For $a \in A$,

$$\boxed{(\text{step}_a h)(s) = h(\delta(a, s))}.$$

Thus state transition acts by precomposition on state-indexed continuations.

For the read operation,

$$\boxed{\text{get}((h_i)_{i \in O})(s) = h_{o(s)}(s)}.$$

Thus output selects among continuations without moving the state.

This gives a useful dual view of a Moore machine. On states, δ evolves the machine forward. On functions out of states, it acts backwards by pullback. Meanwhile o partitions or labels the state space and therefore decides which observable continuation is active.

7.2 Executing a handler against the Moore machine

Let B be a T_{Moore} -algebra. Write its unary operations as

$$u_a : B \rightarrow B$$

and its reader operation as

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

Then $B \otimes_T S$ is obtained from $B \times S$ by the execution relations

$$\boxed{(u_a(b), s) \sim (b, \delta(a, s))}$$

and

$$\boxed{(g((b_i)_{i \in O}), s) \sim (b_{o(s)}, s)}.$$

The first relation says that an abstract step in the handler can be executed by advancing the Moore state. The second says that an abstract read can be executed by consulting the Moore output and selecting the corresponding branch.

The adjunction is

$$\boxed{\mathbf{Set}(B \otimes_T S, X) \cong \mathbf{Alg}(T_{\text{Moore}})(B, X^S)}.$$

The right side gives an X -valued semantics of the abstract handler in functions on machine states. The left side first runs the abstract operations against the machine and then assigns X -values to the resulting execution classes. These are two descriptions of the same information.

7.3 The Moore machine as an interpreter

The same runner can be seen directly as an interpreter

$$\theta_X : T_{\text{Moore}}X \times S \longrightarrow X \times S.$$

When it encounters step_a , it updates

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

When it encounters get , it chooses the branch indexed by $o(s)$, without changing s . When it reaches a returned value x , it outputs (x, s') , where s' is the final runner state.

The monad laws for T_{Moore} , together with the comodel laws, express the compositionality of this interpretation: interpreting a substituted computation is the same as interpreting the outer computation and then continuing with the interpreted inner computation.

8 Where we have got to

It is worth putting the sequence in one table.

Theory	Runner/comodel structure	What it means
$T = R = \text{Id}$	a set S	no dynamics yet
$\mathbb{N}y, \text{Id}$	$f : S \rightarrow S$	one abstract step advances a clock/state
$\mathbb{N}y, \mathbb{N}y$	$f : S \rightarrow S, k : S \rightarrow \mathbb{N}$	one dynamics realised by a state-dependent number of steps of another
$F(y^O), \text{Id}$	$S \rightarrow O \times S$	stateful branch selection
y^O, Id	$o : S \rightarrow O$	pure observation
$T_{\text{Moore}}, \text{Id}$	$\delta : A \times S \rightarrow S, o : S \rightarrow O$	ordinary Moore machine

The common pattern is now visible. Syntax on the T -side becomes costructure on the runner side. Quotient equations on the theory become coherence conditions on what the runner is allowed to do. The runner then acts on handlers by exponentiating over its state space and interpreting the abstract operations there.

9 What comes next: residual effects

So far, with $R = \text{Id}$, the runner has completely handled every abstract effect itself. The next step is the genuinely Sweedler one: allow a residual effect to remain.

For example, with a lottery/randomness monad rand , a runner would have maps

$$\theta_X : T_{\text{Moore}}X \times S \longrightarrow \text{rand}(X \times S).$$

An abstract deterministic Moore operation could then be realised by a probabilistic lower-level mechanism, provided the resulting runner still satisfies the equations of the abstract theory.

This is the conceptual jump from

comodel = complete implementation

to

residual runner = implementation that may delegate effects to R .

That will be the natural starting point for Part II.

References and orientation

The abstract facts used here are standard in the runner/comodel and Sweedler-monad literature. Two particularly useful references are:

- T. Uustalu, *Stateful Runners of Effectful Computations*, Electronic Notes in Theoretical Computer Science 319 (2015), 403–421. The paper develops stateful runners for algebraic effects and shows, for Lawvere theories, that stateful runners are comodels.
- D. McDermott, E. Rivas, and T. Uustalu, *Sweedler Theory of Monads*, FoSSaCS 2022, LNCS 13242, 428–448. In particular, an R -residual stateful runner with carrier S is a family $TX \times S \rightarrow R(X \times S)$, and such a runner induces a functor $\mathbf{Alg}(R) \rightarrow \mathbf{Alg}(T)$ whose underlying carrier functor is $S \Rightarrow -$. The Sweedler hom $\mathcal{M}(T, R)$ is the comonad classifying these runners under the stated accessibility hypotheses.

For the purposes of this tutorial, the main lesson is not a theorem name but a picture:

$$\boxed{\text{abstract effect theory } T \xrightarrow{\text{runner state } S} \text{residual effect theory } R,}$$

and the induced transformation of handlers is carried by functions out of the runner state,

$$A \mapsto A^S.$$

The tensor on the other side of the adjunction is where the concrete execution quotient appears.