Logical Methods in Computer Science
Volume 21, Issue 3, 2025, pp. 18:1–18:52
https://lmcs.episciences.org/

Submitted Jan. 02, 2023
Published Aug. 12, 2025

COINDUCTIVE STREAMS IN MONOIDAL CATEGORIES

ELENA DI LAVORE [a,b,c], GIOVANNI DE FELICE [c,d], AND MARIO ROMÁN [a,c]

[a] Tallinn University of Technology, Ehitajate tee 5, 12616 Tallinn, Estonia

[b] Università di Pisa, Lungarno Antonio Pancinotti 43, 56126 Pisa, Italy

[c] University of Oxford, Oxford OX1 2JD, United Kingdom
e-mail address: elena.di.lavore@cs.ox.ac.uk, mario.roman.garcia@cs.ox.ac.uk

[d] Quantinuum, Oxford OX1 2JD, United Kingdom
e-mail address: giovanni.defelice@cambridgequantum.com

Abstract. We introduce monoidal streams. Monoidal streams are a generalization of causal stream functions, which can be defined in cartesian monoidal categories, to arbitrary symmetric monoidal categories. In the same way that streams provide semantics to dataflow programming with pure functions, monoidal streams provide semantics to dataflow programming with arbitrary processes. Monoidal streams also form a feedback monoidal category. In the same way that the coinductive stream calculus reasons about signal flow graphs, coinductive string diagrams reason about feedback monoidal categories, with semantics in monoidal streams. We highlight the probabilistic case, characterizing probabilistic streams as causal stochastic processes.

Key words and phrases: Monoidal stream, Stream, Monoidal category, Dataflow programming, Feedback, Signal flow graph, Coalgebra, Stochastic process.

The present is an extended version of "Monoidal Streams for Dataflow Programming", by the same authors. Elena Di Lavore and Mario Román were supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001). Mario Román was supported by the Air Force Office of Scientific Research (AFOSR) award number FA9550-21-1-0038. Elena Di Lavore and Mario Román were supported by the Advanced Research + Invention Agency (ARIA) Safeguarded AI Programme.

LOGICAL METHODS IN COMPUTER SCIENCE

DOI:10.46298/LMCS-21(3:18)2025

Contents

1. Introduction

Dataflow languages. Dataflow (or *stream-based*) programming languages, such as Lucid [WA$^+$85, HLR92], follow a paradigm in which every declaration represents an infinite list of values: a *stream* [BCLH93, UV05]. The following program in a Lucid-like language (Figure 1) computes the Fibonacci sequence using the Fby ("followed by") and Wait ("wait") operators: Fby constructs a stream from a first element and a stream; Wait delays a stream by one unit of time.

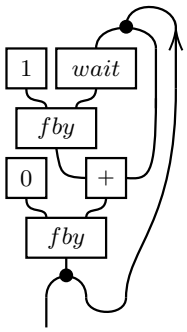
$$\mathit{fib} = 0 \text{ Fby } (\mathit{fib} + (1 \text{ Fby Wait}(\mathit{fib})))$$

Figure 1. *The Fibonacci sequence is 0 followed by the Fibonacci sequence plus the Fibonacci sequence preceded by a 1.*

The control structure of dataflow programs is inspired by *signal flow graphs* [BCLH93, Sha42, Mas53]. Signal flow graphs are diagrammatic specifications of processes with feedback loops, widely used in control system engineering. In a dataflow program, feedback loops represent how the current value of a stream may depend on its previous values. For instance, the previous program (Figure 1) corresponds to the signal flow graph in Figure 2, left.

Monoidal categories. Any theory of processes that *compose sequentially and in parallel*, satisfying reasonable axioms, forms a *monoidal category*. Examples include functions [Lam86], probabilistic channels [CJ19, Fri20], partial maps [RR88, CL02], database queries [BSS18], linear resource theories [CFS16] and quantum processes [AC09]. Signal flow graphs are the graphical syntax for *feedback monoidal categories* [KSW02, BSZ14, BSZ15, DLGR$^+$21, GKS22]: they are the *string diagrams* for any of these theories extended with *feedback*. For instance, the previous program (Figure 1) corresponds to the formal morphism in Figure 2.

Yet, semantics of dataflow languages have been mostly restricted to theories of pure functions [BCLH93, UV08, Cou19, Del19, Oli84]: what are called *cartesian* monoidal categories. We claim that this restriction is actually inessential; dataflow programs may take semantics in non-cartesian monoidal categories, exactly as their signal flow graphs do.



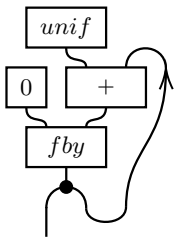
$$\begin{aligned}\mathsf{fib} = \mathbf{fbk}(&\mathsf{copy}\fatsemi\\ &\partial(1 \times \mathsf{wait}) \times \mathrm{id}\fatsemi\\ &\partial(\mathsf{fby}) \times \mathrm{id}\fatsemi\\ &\partial(+)\fatsemi\\ &0 \times \mathrm{id}\fatsemi\\ &\mathsf{fby}\fatsemi\\ &\mathsf{copy})\end{aligned}$$

FIGURE 2. Fibonacci: signal flow graph and morphism.

The present work provides this missing semantics: we construct *monoidal streams* over a symmetric monoidal category, which form a *feedback monoidal category*. Monoidal streams model the values of a *monoidal dataflow language*, in the same way that streams model the values of a classical dataflow language. This opens the door to stochastic, effectful, or quantum dataflow languages. In particular, we give semantics and string diagrams for a *stochastic dataflow programming language*, where the following code can be run (Figure 3).

$$walk = 0 \textsc{ Fby } (\textsc{Uniform}(-1, 1) + walk)$$

FIGURE 3. *A stochastic dataflow program. A random walk is 0 followed by the random walk plus a stochastic stream of steps to the left (-1) or to the right (1), sampled uniformly.*



$$\begin{aligned}\mathsf{walk} = \mathbf{fbk}(&\\ &\partial(\mathsf{unif}) \otimes \mathrm{id}\fatsemi\\ &0 \otimes \partial(+)\fatsemi\\ &\mathsf{fby}\fatsemi\\ &\mathsf{copy})\end{aligned}$$

FIGURE 4. Random walk: signal flow graph and morphism.

1.1. **Contributions.** Our main novel contribution is the definition of a feedback monoidal category of *monoidal streams* over a symmetric monoidal category (Stream, Definition 7.1 and Theorem 7.26). Monoidal streams form a final coalgebra; for sufficiently well-behaved monoidal categories (Definition 6.11), we give an explicit construction of this coalgebra (Definition 6.8).

In cartesian categories, the *causal functions* of Uustalu and Vene [UV05] (see also [SJ19]) are a particular case of our monoidal streams (Theorems 8.6 and 8.9). In the category of stochastic functions, our construction captures the notion of *controlled stochastic process* [FR75, Ros96] (Theorem 9.19).

In order to arrive to this definition, we unify the previous literature: we characterize the cartesian "intensional stateful sequences" of Sprunger and Katsumata [SK19] with a final coalgebra (Theorem 3.5), and then "dinatural stateful sequences" in terms of the

"feedback monoidal categories" of Katis, Sabadini and Walters [KSW02] (Theorem 5.9). We justify observational equivalence with a refined fixpoint equation that employs *coends* (Theorem 6.19). We strictly generalize "stateful sequences" from the cartesian to the monoidal case.

1.2. **Related work.** The theory of monoidal streams that we present in this manuscript merges multiple research threads: feedback and traced monoidal categories, morphism sequences, coalgebra, and categorical dataflow programming. We briefly summarize here these different threads.

Stream programming. The theory of stream programming originates in the '70s with the work of Kahn [Kah74, KM77], following the development of dataflow languages [Fos72, DFL72, Den74, Wen75]. During this period, the interest in dataflow languages is widespread, leading to languages like Lucid and Lustre [Kos73, AW77, WA$^+$85, HLR92, Ack79].

Coalgebraic streams. Uustalu and Vene [UV05] provide elegant *comonadic* semantics for a (cartesian) Lucid-like programming language. We shall prove that their exact technique cannot be possibly extended to arbitrary monoidal categories (Theorem 8.6). However, we recover their semantics as a particular case of our monoidal streams (Theorem 8.9).

Feedback. Feedback monoidal categories are a weakening of *traced monoidal categories*. The construction of the free such categories is originally due to Katis, Sabadini and Walters [KSW02]. Feedback monoidal categories and their free construction have been repurposed and rediscovered multiple times in the literature [KSW97, HMH14, BHP$^+$19, GN03]. Di Lavore et al. [DLGR$^+$21] summarize these uses and introduce *delayed feedback*.

Cartesian stateful sequences. Sprunger and Katsumata constructed the category of *stateful sequences* in the cartesian case [SK19]. The present work extends their ideas to the monoidal case.

Sprunger and Katsumata first introduced the idea of a *delayed trace operator* with a *time-shifting functor*. Its type differs slightly from our *feedback operator* (Definition 5.1) which follows Katis, Sabadini and Walters [KSW02]: it adds a "point", and a forward operator is applied to the other side of the morphism. However, its purpose and essence is similar: we take care that both operators are interderivable in the cartesian case (Remark 8.11).

Sprunger and Katsumata also introduced for the first time the idea of equality by truncation of infinite sequences, and called it "dinatural equivalence". We shall define a similar *truncation up to a future* and prove it is actually the *observational equivalence* of a final coalgebra. Happily, this still particularizes to dinatural equality in the sense of Sprunger and Katsumata when a cartesian structure is present (Theorem 8.9). *Sliding*, which appeared first as a consequence of truncation in the work of Sprunger and Katsumata ("Shim lemma" in [SK19]) can be now, for the first time, the fundamental notion of *dinatural equivalence* (Definition 4.5).

Finally, there is an unfortunate clash of terminology: we use $\mathsf{St}(\bullet)$ for the free category with feedback in the sense of Katis, Sabadini and Walters [KSW97, HMH14, BHP$^+$19]. Theorem 5.9 relates this $\mathsf{St}(\bullet)$ to the work of Sprunger and Katsumata, who use $\mathrm{St}(\bullet)$ for the cartesian analogue of our Stream construction.

Monoidal stateful sequences. Our work is based and significantly expands an unpublished work by Román [Rom20], who first exteneded Sprunger and Katsumata's [SK19] definition to the symmetric monoidal case, using *coends* to justify *dinatural equality*. Shortly after, Carette, de Visme and Perdrix [CdVP21] rederived this construction and applied it to the case of completely positive maps between Hilbert spaces, using (a priori) a slightly different notion of equality. We synthetise this previous work on monoidal sequences, we justify it for the first time using coalgebra and we particularize it to some cases of interest.

General and dependent streams. Our work concerns *synchronous* streams: those where, at each point in time $t = 0, 1, \ldots$, the stream process takes exactly one input and produces exactly one output. This condition is important in certain contexts like, for instance, real-time embedded systems; but it is not always present. The study of asynchronous stream transformers and their universal properties is considerably different [AI15], and we refer the reader to the recent work of Garner [Gar21] for a discussion on *non-synchronous* streams. Finally, when we are concerned with *dependent streams* indexed by time steps, a possible approach, when our base category is a topos, is to use the *topos of trees* [BMSS11].

Categorical dataflow programming. Category theory is a common tool of choice for dataflow programming [Rut00, GN03, Mam20]. In particular, profunctors and coends are used by Hildebrandt, Panangaden and Winskel [HPW98] to generalise a model of non-deterministic dataflow, which has been the main focus [PS88, LS89, LM09] outside cartesian categories.

1.3. **Synopsis.** This manuscript contains three main definitions in terms of universal properties (*intensional, dinatural and observational streams*, Definitions 3.1, 6.2 and 5.8); and three explicit constructions for them (*intensional, dinatural and observational sequences*, Definitions 3.4, 4.5 and 6.8). Each definition is refined into the next one: each construction is a quotienting of the previous one.

Sections 2.1 and 4.1 contain expository material on coalgebra and dinaturality. Section 3 presents intensional monoidal streams. Section 5 introduces dinatural monoidal streams in terms of feedback monoidal categories. Section 6 introduces the definitive *observational* equivalence and defines *monoidal streams*. Section 7 constructs the feedback monoidal category of monoidal streams. Sections 8 and 9 present two examples: cartesian and stochastic streams.

2. Coalgebra & Monoidal Categories

2.1. **Coalgebra.** In this preparatory section, we introduce some background material on coalgebra [Rut00, Jac16, Adá05]. Coalgebra is the category-theoretic study of stateful systems and infinite data-structures, such as streams. These structures arise as *final coalgebras*: universal solutions to certain functor equations.

Let us fix an endofunctor $F \colon \mathsf{C} \to \mathsf{C}$ through the section.

Definition 2.1. A *coalgebra* (Y, β) is an object $Y \in \mathsf{C}$, together with a morphism $\beta\colon Y \to FY$. A *coalgebra morphism* $g\colon (Y,\beta) \to (Y',\beta')$ is a morphism $g\colon Y \to Y'$ such that $g \fatsemi \beta' = \beta \fatsemi Fg$.

$$\begin{array}{ccc} Y & \xrightarrow{g} & Y' \\ {\scriptstyle\beta}\downarrow & & \downarrow{\scriptstyle\beta'} \\ FY & \xrightarrow{Fg} & FY' \end{array}$$

Coalgebras for an endofunctor form a category with coalgebra morphisms between them. A *final coalgebra* is a final object in this category. As such, final coalgebras are unique up to isomorphism when they exist.

Definition 2.2. A *final coalgebra* is a coalgebra (Z, γ) such that for any other coalgebra (Y, β) there exists a unique coalgebra morphism $g\colon (Y,\beta) \to (Z,\gamma)$.

Our interest in final coalgebras derives from the fact that they are canonical fixpoints of an endofunctor. Specifically, Lambek's theorem (Theorem 2.4) states that whenever the final coalgebra exists, it is a fixpoint.

Definition 2.3. A *fixpoint* is a coalgebra (Y, β) such that $\beta\colon Y \to FY$ is an isomorphism. A *fixpoint morphism* is a coalgebra morphism between fixpoints: fixpoints and fixpoint morphisms form a full subcategory of the category of coalgebras. A *final fixpoint* is a final object in this category.

Theorem 2.4 (Lambek, [Lam68])**.** *Final coalgebras are fixpoints. As a consequence, when they exist, they are final fixpoints.*

The last question before continuing is how to explicitly construct a final coalgebra. This is answered by Adamek's theorem (Theorem 2.5). The reader may be familiar with Kleene's theorem for constructing fixpoints [SLG94]: the least fixpoint of a monotone function $f\colon X \to X$ in a directed-complete partial order $(X, \leqslant)$ is the supremum of the chain $\bot \leqslant f(\bot) \leqslant f(f(\bot)) \leqslant \dots$, where $\bot$ is the least element of the partial order, whenever this supremum is preserved by f. This same result can be categorified into a fixpoint theorem for constructing final coalgebras: the directed-complete poset becomes a category with ω-chain limits; the monotone function becomes an endofunctor; and the least element becomes the final object.

Theorem 2.5 (Adamek, [Ada74])**.** *Let* D *be a category with a final object* 1 *and* ω^{op}*-shaped limits. Let* $F\colon \mathsf{D} \to \mathsf{D}$ *be an endofunctor. We write* $L = \lim_n F^n 1$ *for the limit of the following* ω*-chain, which is called the* terminal sequence *of* F*.*

$$1 \xleftarrow{!} F1 \xleftarrow{F!} FF1 \xleftarrow{FF!} FFF1 \xleftarrow{FFF!} \dots$$

Assume that F *preserves this limit, meaning that the canonical morphism* $FL \to L$ *is an isomorphism. Then,* L *is the final* F*-coalgebra.*

2.2. **Monoidal categories.** In order to fix notation, we include background material on monoidal categories. Monoidal categories have a sound and complete syntax in terms of string diagrams, which is the one we will mostly use during this text [JS91]. However, when proving that some category is indeed monoidal, we may prefer the classical definition of monoidal categories.

Definition 2.6 [Mac78]**.** A **monoidal category**,

$$(\mathsf{C}, \otimes, I, \alpha, \lambda, \rho),$$

is a category C equipped with a functor $\otimes\colon \mathsf{C} \times \mathsf{C} \to \mathsf{C}$, a unit $I \in \mathsf{C}$, and three natural isomorphisms: the associator $\alpha_{A,B,C}\colon (A \otimes B) \otimes C \cong A \otimes (B \otimes C)$, the left unitor $\lambda_A\colon I \otimes A \cong A$ and the right unitor $\rho_A\colon A \otimes I \cong A$; such that $\alpha_{A,I,B} \fatsemi (\mathrm{id}_A \otimes \lambda_B) = \rho_A \otimes \mathrm{id}_B$ and $(\alpha_{A,B,C} \otimes \mathrm{id}) \fatsemi \alpha_{A,B\otimes C,D} \fatsemi (\mathrm{id}_A \otimes \alpha_{B,C,D}) = \alpha_{A\otimes B,C,D} \fatsemi \alpha_{A,B,C\otimes D}$. A monoidal category is *strict* if α, λ and ρ are identities.

Definition 2.7 (Monoidal functor, [Mac78])**.** Let

$$(\mathsf{C}, \otimes, I, \alpha^{\mathsf{C}}, \lambda^{\mathsf{C}}, \rho^{\mathsf{C}}) \text{ and } (\mathsf{D}, \boxtimes, J, \alpha^{\mathsf{D}}, \lambda^{\mathsf{D}}, \rho^{\mathsf{D}})$$

be monoidal categories. A *monoidal functor* (sometimes called *strong monoidal functor*) is a triple (F, ε, μ) consisting of a functor $F\colon \mathsf{C} \to \mathsf{D}$ and two natural isomorphisms $\varepsilon\colon J \cong F(I)$ and $\mu\colon F(A \otimes B) \cong F(A) \boxtimes F(B)$; such that the associators satisfy the *pentagon equation*

$$\alpha^{\mathsf{D}}_{FA,FB,FC} \fatsemi (\mathrm{id}_{FA} \otimes \mu_{B,C}) \fatsemi \mu_{A,B\otimes C} = (\mu_{A,B} \otimes \mathrm{id}_{FC}) \fatsemi \mu_{A\otimes B,C} \fatsemi F(\alpha^{\mathsf{C}}_{A,B,C}),$$

and the unitors satisfy $(\varepsilon \otimes \mathrm{id}_{FA}) \fatsemi \mu_{I,A} \fatsemi F(\lambda^{\mathsf{C}}_A) = \lambda^{\mathsf{D}}_{FA}$ and $(\mathrm{id}_{FA} \otimes \varepsilon) \fatsemi \mu_{A,I} \fatsemi F(\rho^{\mathsf{C}}_{FA}) = \rho^{\mathsf{D}}_{FA}$.

A monoidal functor is a *monoidal equivalence* if it is moreover an equivalence of categories. Two monoidal categories are monoidally equivalent if there exists a monoidal equivalence between them.

During most of the paper, we omit all associators and unitors from monoidal categories, implicitly using the *coherence theorem* for monoidal categories (Remark 2.9).

Theorem 2.8 (Coherence theorem, [Mac78])**.** *Every monoidal category is monoidally equivalent to a strict monoidal category.*

Remark 2.9. Let us comment further on how we use the coherence theorem. Each time we have a morphism $f\colon A \to B$ in a monoidal category, we have a corresponding morphism $A \to B$ in its strictification. This morphism can be lifted to the original category to uniquely produce, say, a morphism $(\lambda_A \fatsemi f \fatsemi \lambda_B^{-1})\colon I \otimes A \to I \otimes B$. Each time the source and the target are clearly determined, we simply write f again for this new morphism.

Instead of dealing with arbitrary monoidal categories, we mostly work with these that represent theories of processes: *symmetric monoidal categories*, those where resources can be rerouted freely thanks to an self-invertible *symmetry* morphism $\sigma_{A,B}\colon A \otimes B \to B \otimes A$.

Definition 2.10 (Symmetric monoidal category, [Mac78])**.** A *symmetric monoidal category* $(\mathsf{C}, \otimes, I, \alpha, \lambda, \rho, \sigma)$ is a monoidal category $(\mathsf{C}, \otimes, I, \alpha, \lambda, \rho)$ equipped with a symmetry $\sigma_{A,B}\colon A \otimes B \to B \otimes A$, which satisfies the hexagon equation

$$\alpha_{A,B,C} \fatsemi \sigma_{A,B\otimes C} \fatsemi \alpha_{B,C,A} = (\sigma_{A,B} \otimes \mathrm{id}) \fatsemi \alpha_{B,A,C} \fatsemi (\mathrm{id} \otimes \sigma_{A,C})$$

and additionally satisifes $\sigma_{A,B} \fatsemi \sigma_{B,A} = \mathrm{id}$.

Remark 2.11 (Notation)**.** String diagrams will be drawn top to bottom. We write $\underline{a}$ for the morphism a tensored with some identities when these can be deduced from the context; we write σ for any symmetry, and only omit it when it can be deduced from the context. For instance, let $f\colon A \to B$, let $h\colon B \to D$ and let $g\colon C \otimes D \to E$. We write $\underline{(f \fatsemi h)} \fatsemi \sigma \fatsemi g\colon A \otimes C \to E$ for the morphism $(f \otimes \mathrm{id}) \fatsemi \sigma \fatsemi (\mathrm{id} \otimes h) \fatsemi g$, which could have been also written as $(f \otimes \mathrm{id}) \fatsemi (h \otimes \mathrm{id}) \fatsemi \sigma \fatsemi g$.

Definition 2.12 [Mac78]**.** A symmetric monoidal functor between two symmetric monoidal categories $(\mathsf{C}, \sigma^{\mathsf{C}})$ and $(\mathsf{D}, \sigma^{\mathsf{D}})$ is a monoidal functor $F\colon \mathsf{C} \to \mathsf{D}$ such that $\sigma^{\mathsf{D}} \fatsemi \mu = \mu \fatsemi F(\sigma^{\mathsf{C}})$.

Definition 2.13. A *cartesian monoidal category* is a monoidal category whose tensor is the categorical product and whose unit is a terminal object.

2.3. **Size concerns, limits and colimits.** We call Set the category of sets and functions below a certain Grothendieck universe [ML69]. We do take colimits (and coends) over this category without creating size issues: we can be sure of their existence in our metatheoretic category of sets.

Proposition 2.14 (from [Ada19, Theorem 13])**.** *Terminal coalgebras exist in* $\mathsf{Set}_{\leqslant\lambda}$*. More precisely, the category of sets below a certain regular uncountable cardinal* λ *is algebraically complete and cocomplete; meaning that every* Set*-endofunctor has a terminal coalgebra and an initial algebra.*

A connected category is an inhabited category whose underlying graph is connected. A connected limit is the limit of a diagram $\mathbf{D}\colon \mathsf{J} \to \mathsf{C}$ whose domain category J is connected. The definition of intensional sequences (Section 3.2) relies on connected limits, and their characterisation on the following folklore result.

Theorem 2.15 (Coproducts commute with connected limits)**.** *Let* I *be a set, understood as a discrete category, and let* A *be a connected category with* $F\colon I \times \mathsf{A} \to \mathsf{Set}$ *a functor. The canonical morphism*

$$\sum_{i\in I} \lim_{a\in A} F(i,a) \to \lim_{a\in A} \sum_{i\in I} F(i,a)$$

is an isomorphism.

In particular, let $F_n\colon I \to \mathsf{Set}$ *be a family of functors indexed by the natural numbers with a family of natural transformations* $\alpha_n\colon F_{n+1} \to F_n$*. The canonical morphism*

$$\sum_{i\in I} \lim_{n\in\mathbb{N}} F_n(i) \to \lim_{n\in\mathbb{N}} \sum_{i\in I} F_n(i)$$

is an isomorphism.

Proof. Note that there are no morphisms between any two indices $i, j \in I$. Once some $i \in I$ is chosen in any factor of the connected limit, it forces any other factor to also choose $i \in I$. This makes the local choice of $i \in I$ be equivalent to the global choice of $i \in I$. □

3. Intensional Monoidal Streams

This section introduces a preliminary definition of *monoidal stream* in terms of a fixpoint equation (in Figure 5). In later sections, we refine both this definition and its characterization into the definitive notion of *monoidal stream*.

We fix a small symmetric monoidal category $(\mathsf{C}, \otimes, I)$.

3.1. **The fixpoint equation.** Classically, type-variant streams have a neat coinductive definition [Jac16, Rut00] that says:

> *"A stream of type* $\mathbb{A} = (A_0, A_1, \dots)$ *is an element of* A_0 *together with a stream of type* $\mathbb{A}^+ = (A_1, A_2, \dots)$*".*

Formally, the set of streams is the final fixpoint of the equation

$$\mathbf{S}(A_0, A_1, \dots) \cong A_0 \times \mathbf{S}(A_1, A_2, \dots);$$

and this fixpoint is computed to be $\mathbf{S}(\mathbb{A}) = \prod_{n\in\mathbb{N}}^{\infty} A_n$. In the same vein, we want to introduce not only streams but *stream processes* over a fixed theory of processes.

> *"A stream process from* $\mathbb{X} = (X_0, X_1, \dots)$ *to* $\mathbb{Y} = (Y_0, Y_1, \dots)$ *is a process from* X_0 *to* Y_0 *communicating along a channel* M *with a stream process from* $\mathbb{X}^+ = (X_1, X_2, \dots)$ *to* $\mathbb{Y}^+ = (Y_1, Y_2, \dots)$*."*

Streams are recovered as stream processes on an empty input, so we take this more general slogan as our preliminary definition of *monoidal stream* (in Definition 3.1). Formally, they are the final fixpoint of the equation in Figure 5.

Definition 3.1. The set of *intensional monoidal streams* $\mathbf{T}\colon [\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}] \to \mathsf{Set}$, depending on inputs and outputs, is the final fixpoint of the equation in Figure 5.

$$\mathbf{T}(\mathbb{X}, \mathbb{Y}) \cong \sum_{M\in\mathsf{C}} \hom(X_0, M \otimes Y_0) \times \mathbf{T}(M \cdot \mathbb{X}^+, \mathbb{Y}^+).$$

FIGURE 5. Fixpoint equation for intensional streams.

Remark 3.2 (Notation)**.** Let $\mathbb{X} \in [\mathbb{N}, \mathsf{C}]$ be a sequence of objects $(X_0, X_1, \dots)$. We write $\mathbb{X}^+$ for its *tail* $(X_1, X_2, \dots)$. Given $M \in \mathsf{C}$, we write $M \cdot \mathbb{X}$ for the sequence $(M \otimes X_0, X_1, X_2, \dots)$; As a consequence, we write $M \cdot \mathbb{X}^+$ for $(M \otimes X_1, X_2, X_3, \dots)$.

Remark 3.3 (Initial fixpoint)**.** There exists an obvious fixpoint for the equation in Figure 5: the constant empty set. This solution is the *initial* fixpoint, a minimal solution. The *final* fixpoint will be realized by the set of *intensional sequences*.

3.2. **Intensional sequences.** We now construct the set of intensional streams explicitly (Theorem 3.5). For this purpose, we generalize the "stateful morphism sequences" of Sprunger and Katsumata [SK19] from cartesian to arbitrary symmetric monoidal categories (Definition 3.4). We derive a novel characterization of these "sequences" as the desired final fixpoint (Theorem 3.5).

In the work of Sprunger and Katsumata, a stateful sequence is a sequence of morphisms $f_n\colon M_{n-1} \times X_n \to M_n \times Y_n$ in a cartesian monoidal category. These morphisms represent a process at each point in time $n = 0, 1, 2, \dots$. At each step n, the process takes an input X_n and, together with the stored memory M_{n-1}, produces some output Y_n and writes to a new memory M_n. The memory is initially empty, with $M_{-1} = \mathbf{1}$ being the final object by convention. We extend this definition to any symmetric monoidal category.

Definition 3.4. Let $\mathbb{X}$ and $\mathbb{Y}$ be two sequences of objects representing inputs and outputs, respectively. An *intensional sequence* is a sequence of objects $(M_0, M_1, \dots)$ together with a sequence of morphisms

$$(f_n\colon M_{n-1} \otimes X_n \to M_n \otimes Y_n)_{n\in\mathbb{N}},$$

where, by convention, $M_{-1} = I$ is the unit of the monoidal category. In other words, the set of intensional sequences is

$$\mathbf{Int}(\mathbb{X}, \mathbb{Y}) \coloneqq \sum_{M \in [\mathbb{N}, \mathsf{C}]} \prod_{n=0}^{\infty} \hom(M_{n-1} \otimes X_n, M_n \otimes Y_n).$$

We now prove that intensional sequences are the final fixpoint of the equation in Figure 5. The following Theorem 3.5 serves two purposes: it gives an explicit final solution to this fixpoint equation and it gives a novel universal property to intensional sequences.

Theorem 3.5. *Intensional sequences are the explicit construction of intensional streams,* $\mathbf{T} \cong \mathbf{Int}$*. In other words, they are a fixpoint of the equation in Figure 5, and they are the final such one.*

Proof. Let us first see that our candidate, Expression 3.1, is indeed a fixpoint of Equation 5.

$$\sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \prod_{i=0}^{\infty} \hom(M_{i-1} \otimes X_i; M_i \otimes Y_i). \tag{3.1}$$

For this, we note that cartesian products distribute over coproducts: the following Expression 3.2 is isomorphic to Expression 3.1. This proves it is a fixpoint.

$$\sum_{M_0} \hom(X_0, M_0 \otimes Y_0) \times \sum_{M \in [\mathbb{N}, \mathsf{C}]} \prod_{n=1}^{\infty} \hom(M_{n-1} \otimes X_n, M_n \otimes Y_n). \tag{3.2}$$

If Expression 3.1 were to coincide with Expression 3.3, our desired result would follow by Adamek's theorem (Theorem 2.5). Adamek's theorem states that, if the following limit exists and is a fixpoint, then it is indeed the final fixpoint. We already know that it must exist: categories of functors over sets, such as $[[\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}], \mathsf{Set}]$, have all limits.

$$\lim_{n \in \mathbb{N}} \sum_{M_0, \dots, M_n} \prod_{t=0}^{n} \hom(M_{t-1} \otimes X_t, M_t \otimes Y_t) \tag{3.3}$$

Therefore, it suffices to prove that Expression 3.1 and Expression 3.3 are isomorphic.

Let us prove this isomorphism. For convenience, we call the main factor of this limit $\Omega_i(M_i, M_{i-1}) = \hom(M_{i-1} \otimes X_i; M_i \otimes Y_i)$. Let us first note that the limit in Equation (3.1) can be simplified, using that (i) limits commute with limits and that (ii) connected limits commute with coproducts.

$$\sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \prod_{i=0}^{\infty} \Omega_i(M_i, M_{i-1}) \overset{(i)}{\cong} \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \lim_{k \in \mathbb{N}} \prod_{i=0}^{k} \Omega_i(M_i, M_{i-1}) \overset{(ii)}{\cong} \lim_{k \in \mathbb{N}} \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \prod_{i=0}^{k} \Omega_i(M_i, M_{i-1}).$$

Again for convenience, we now call $\Psi_k(M_0, \dots, M_k) = \prod_{i=0}^{k} \Omega_i(M_{i-1}, M_i)$. We will use that the coproduct over mute variables can be read as a product, $\sum_{a \in A} B \cong A \times B$, to simplify each one of the factors of the limit. For each $k \in \mathbb{N}$, we have that

$$\sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_k(M_0, \dots, M_k) \cong \left(\prod_{j=k_n}^{\infty} \mathbb{C}_{\text{obj}} \right) \times \sum_{M_0, \dots, M_k} \Psi_k(M_0, \dots, M_k).$$

As a result, we can compute the following limit.

$$\lim_k \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_k(M_0, \ldots, M_k) \cong \lim_k \left(\prod_{j=k_n}^{\infty} \mathbb{C}_{\text{obj}} \right) \times \sum_{M_0, \ldots, M_k} \Psi_k(M_0, \ldots, M_k).$$

Under this isomorphism, the limit diagram gets transformed into a more tractable diagram that we will be able to compute. The original limit diagram consisted only of projections,

$$\pi_k \colon \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_{k+1}(M_0, \ldots, M_{k+1}) \to \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_k(M_0, \ldots, M_k).$$

The diagram under the isomorphism contains objects $\mathbb{C}_{\text{obj}}^{(k,j)}$ for each $j > k$, and objects $\sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_k(M_0, \ldots, M_k)$ for each $k \in \mathbb{N}$; let us use $\Sigma\Psi_n$ as a shorthand for this term. The diagram contains three families of arrows: the projections between sums, $\alpha_k \colon \Sigma\Psi_{k+1} \to \Sigma\Psi_k$ the projections out of sums, $\beta_k \colon \Sigma\Psi_{k+1} \to \mathbb{C}^{(k,k+1)}$, and projections between objet choices, $\gamma_{k,j} \colon \mathbb{C}^{(k+1,j)} \to \mathbb{C}^{(k,j)}$. See the right hand side of the following Figure 6.

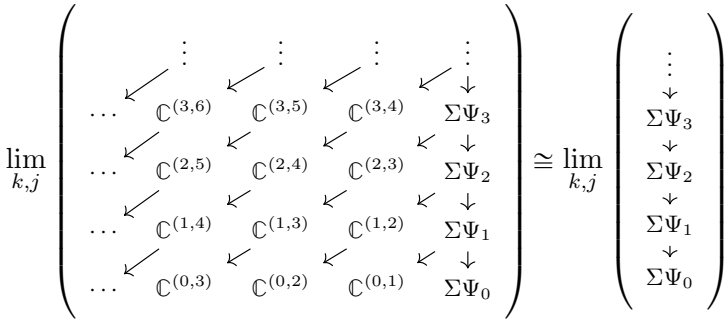


FIGURE 6. Simplification of a limit with an initial subdiagram.

Finally, we can use the fact that the limit of a diagram coincides with the limit of any of its dominating chains [Mac78, Theorem IX.3.1]. In other words, the chain of projections on the left hand side of Figure 6 is initial, and we can conclude that

$$\lim_k \sum_{M_n \in [\mathbb{N}, \mathsf{C}]} \Psi_k(M_0, \ldots, M_k) \;\cong\; \lim_k \sum_{M_0, \ldots, M_k} \Psi_k(M_0, \ldots, M_k).$$

Putting all these isomorphisms together, we have proven that the two limits in Equation (3.3) and Equation (3.1) are isomorphic. This concludes the proof. □

4. Dinaturality and Coends

4.1. **Interlude: Dinaturality.** During the rest of this text, we deal with two different definitions of what it means for two processes to be equal: *dinatural* and *observational equivalence*, apart from pure *intensional equality*. Fortunately, when working with functors of the form $P \colon \mathsf{C}^{op} \times \mathsf{C} \to \mathsf{Set}$, the so-called *endoprofunctors*, we already have a canonical notion of equivalence.

Endoprofunctors $P\colon \mathsf{C}^{op} \times \mathsf{C} \to \mathsf{Set}$ can be thought as indexing families of processes $P(M,N)$ by the types of an input channel M and an output channel N. A process $p \in P(M,N)$ writes to a channel of type N and then reads from a channel of type M.

Now, assume we also have a transformation $r\colon N \to M$ translating from output to input types. Then, we can *plug the output to the input*: the process p writes with type N, then r translates from N to M, and then p uses this same output as its input M. This composite process can be given two sligthly different descriptions; the process could

- ▷ translate *after writing*, $P(M,r)(p) \in P(M,M)$, or
- ▷ translate *before reading*, $P(r,N)(p) \in P(N,N)$.

These two processes have different types. However, with the output plugged to the input, it does not really matter when to apply the translation. These two descriptions represent the same process: they are *dinaturally equivalent*.

Definition 4.1 (Dinatural equivalence)**.** For any functor $P\colon \mathsf{C}^{op} \times \mathsf{C} \to \mathsf{Set}$, consider the set

$$S_P = \sum_{M \in \mathsf{C}} P(M,M).$$

Dinatural equivalence, $(\sim)$, on the set S_P is the smallest equivalence relation satisfying $P(M,r)(p) \sim P(r,N)(p)$ for each $p \in P(M,N)$ and each $r \in \hom(N,M)$.

Coproducts quotiented by dinatural equivalence construct a particular form of colimit called a *coend*. Under the process interpretation of profunctors, taking a coend means *plugging an output to an input* of the same type.

4.2. **Coend calculus.** *Coend calculus* is the name given to the a branch of category theory that describes the behaviour of certain colimits called *coends*. MacLane [Mac78] and Loregian [Lor21] give introductory presentations of the coend calculus.

Definition 4.2 (Coend, [Mac78, Lor21])**.** Let $P\colon \mathsf{C}^{op} \times \mathsf{C} \to \mathsf{Set}$ be a functor. Its *coend* is the coproduct of $P(M,M)$ indexed by $M \in \mathsf{C}$, quotiented by dinatural equivalence.

$$\int^{M \in \mathsf{C}} P(M,M) = \left(\sum_{M \in \mathsf{C}} P(M,M) \Big/ \sim \right).$$

That is, the coend is the colimit of the diagram with a *cospan* $P(M,M) \leftarrow P(M,N) \to P(N,N)$ for each $f\colon N \to M$.

Proposition 4.3 (Yoneda reduction)**.** *Let* C *be any category and let* $F\colon \mathsf{C} \to \mathsf{Set}$ *be a functor; the following isomorphism holds for any given object* $A \in \mathsf{C}$*.*

$$\int^{X \in \mathsf{C}} \hom(X,A) \times FX \cong FA.$$

Following the analogy with classical analysis, the hom *profunctor works as a Dirac delta.*

Proposition 4.4 (Fubini rule)**.** *Coends commute between them; that is, there exists a natural isomorphism*

$$\int^{X_1 \in \mathsf{C}} \int^{X_2 \in \mathsf{C}} P(X_1, X_2, X_1, X_2) \cong \int^{X_2 \in \mathsf{C}} \int^{X_1 \in \mathsf{C}} P(X_1, X_2, X_1, X_2).$$

In fact, they are both isomorphic to the coend over the product category,

$$\int^{(X_1,X_2)\in\mathsf{C}\times\mathsf{C}} P(X_1,X_2,X_1,X_2).$$

Following the analogy with classical analysis, coends follow the Fubini rule for integrals.

4.3. **Towards dinatural memory channels.** Let us go back to intensional monoidal streams. Consider a family of processes $f_n\colon M_{n-1}\otimes X_n \to Y_n\otimes N_n$ reading from memories of type M_n but writing to memories of type N_n. Assume we also have processes $r_n\colon N_n\to M_n$ translating from output to input memory. Then, we can consider the process that does f_n, translates from memory N_n to memory M_n and then does f_{n+1}. This process is described by two different intensional sequences,

- $(f_n;(r_n\otimes\mathrm{id})\colon M_{n-1}\otimes X_n\to M_n\otimes Y_n)_{n\in\mathbb{N}}$, and
- $((r_{n-1}\otimes\mathrm{id});f_n\colon N_{n-1}\otimes X_n\to N_n\otimes Y_n)_{n\in\mathbb{N}}$.

These two intensional sequences have different types for the memory channels. However, in some sense, they represent *the same process description.* If we do not care about what exactly it is that we save to memory, we should consider two such processes to be equal (as in Figure 7, where "the same process" can keep two different values in memory). Indeed, dinaturality in the memory channels M_n is the smallest equivalence relation $(\sim)$ satisfying

$$(f_n;(r_n\otimes\mathrm{id}))_{n\in\mathbb{N}} \sim ((r_{n-1}\otimes\mathrm{id});f_n)_{n\in\mathbb{N}}.$$

This is precisely the quotienting that we perform in order to define *dinatural sequences.*

Definition 4.5. *Dinatural equivalence* of intensional sequences, $(\sim)$, is dinatural equivalence in the memory channels M_n. A *dinatural sequence* from $\mathbb{X}$ to $\mathbb{Y}$ is an equivalence class

$$\langle f_n\colon M_{n-1}\otimes X\to M_n\otimes Y\rangle_{n\in\mathbb{N}}$$

of intensional sequences under dinatural equivalence.

In other words, the set of dinatural sequences is the set of intensional sequences substituting the coproduct by a coend,

$$\mathbf{Dnt}(\mathbb{X},\mathbb{Y}) = \int^{M\in[\mathbb{N},\mathsf{C}]} \prod_{i=0}^{\infty} \hom(X_i\otimes M_{i-1}, Y_i\otimes M_i).$$

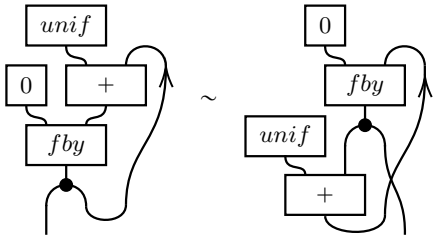


FIGURE 7. Dinaturally equivalent walks keeping different memories: the *current* position vs. the *next* position.

5. Dinatural Monoidal Streams

In this section, we introduce *dinatural monoidal streams* in terms of a universal property: *dinatural streams are the morphisms of the free delayed-feedback monoidal category* (Theorem 5.9).

Feedback monoidal categories come from the work of Katis, Sabadini and Walters [KSW02]. They are instrumental to our goal of describing and composing signal flow graphs: they axiomatize a graphical calculus that extends the well-known string diagrams for monoidal categories with *feedback loops* [KSW02, DLGR$^+$21]. Constructing the *free feedback monoidal category* (Definition 5.5) will lead to the main result of this section: dinatural sequences are the explicit construction of dinatural streams (Theorem 5.9).

We finish the section by exploring how dinatural equivalence may not be enough to capture true observational equality of processes (Example 5.11).

5.1. Feedback monoidal categories.

Feedback monoidal categories are symmetric monoidal categories with a "feedback" operation that connects outputs to inputs. They have a natural axiomatisation (Definition 5.1) that has been rediscovered independently multiple times, with only slight variations [BÉ93, KSW02, KSW99, BHP$^+$19]. Feedback is weaker than *trace* while still satisfying a normalisation property that allows a straightforward construction of the free feedback monoidal category (Theorem 5.6, cf. [DLGR$^+$21]). Traced monoidal categories are better known—yet more restrictive: it is the failure of their yanking axiom what will allow stateful morphisms. We present a novel definition that generalizes the previous ones by allowing the feedback operator to be guarded by a monoidal endofunctor.

Definition 5.1. A *feedback monoidal category* is a symmetric monoidal category $(\mathsf{C}, \otimes, I)$ endowed with a symmetric strong monoidal endofunctor $\mathbf{F}\colon \mathsf{C} \to \mathsf{C}$ and an operation

$$\mathbf{fbk}_S\colon \hom(\mathbf{F}(S) \otimes X, S \otimes Y) \to \hom(X, Y)$$

for all S, X and Y objects of C; this operation needs to satisfy the following axioms.

(A1). Tightening: $u \fatsemi \mathbf{fbk}_S(f) \fatsemi v = \mathbf{fbk}_S((\mathsf{id}_{\mathbf{F}S} \otimes u) \fatsemi f \fatsemi (\mathsf{id}_S \otimes v))$.
(A2). Vanishing: $\mathbf{fbk}_I(f) = f$.
(A3). Joining: $\mathbf{fbk}_T(\mathbf{fbk}_S(f)) = \mathbf{fbk}_{S\otimes T}(f)$
(A4). Strength: $\mathbf{fbk}_S(f) \otimes g = \mathbf{fbk}_S(f \otimes g)$.
(A5). Sliding: $\mathbf{fbk}_S((\mathbf{F}h \otimes \mathsf{id}_X) \fatsemi f) = \mathbf{fbk}_T(f \fatsemi (h \otimes \mathsf{id}_Y))$.

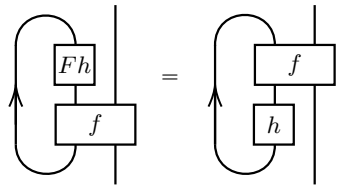


Figure 8. The sliding axiom (A5).

Definition 5.2. A *feedback functor* between two feedback monoidal categories $(\mathsf{C}, \mathbf{F}^{\mathsf{C}}, \mathbf{fbk}^{\mathsf{C}})$ and $(\mathsf{D}, \mathbf{F}^{\mathsf{D}}, \mathbf{fbk}^{\mathsf{D}})$ is a symmetric monoidal functor $G\colon \mathsf{C} \to \mathsf{D}$ such that $G \fatsemi \mathbf{F}^{\mathsf{D}} = \mathbf{F}^{\mathsf{C}} \fatsemi G$ and

$$G(\mathbf{fbk}^{\mathsf{C}}_S(f)) = \mathbf{fbk}^{\mathsf{D}}_{GS}(\mu_{\mathbf{F}S,A} \fatsemi Gf \fatsemi \mu^{-1}_{S,B})$$

for each $f\colon \mathbf{F}S \otimes X \to S \otimes Y$, where $\mu_{A,B}\colon G(A) \otimes G(B) \to G(A \otimes B)$ is the structure morphism of the monoidal functor G.

A traced monoidal category is, in fact, a feedback monoidal category with an extra axiom, the *yanking axiom*, that makes the feedback loop "instantaneous" (Figure 9, left). This requirement imposes that the internal state does not change between iterations and the process has already reached equilibrium: the yanking axiom implies memoryless processes. These conditions are too strong for our purposes of modelling dataflow computations, so we choose feedback monoidal categories instead. For a thorough discussion about the conceptual difference between trace and feedback, and the necessity to drop the yanking axiom for modelling state, see [DLGR$^+$23, Sections 3.2 and 3.3].

Remark 5.3 (Wait or trace)**.** In a feedback monoidal category $(\mathsf{C}, \mathbf{fbk})$, we construct the morphism $\mathsf{wait}_X\colon X \to FX$ as a feedback loop over the symmetry, $\mathsf{wait}_X = \mathbf{fbk}(\sigma_{F(X),X})$ (Figure 9, right). A *traced monoidal category* [JSV96] is a feedback monoidal category guarded by the identity functor such that $\mathsf{wait}_X = \mathrm{id}_X$.

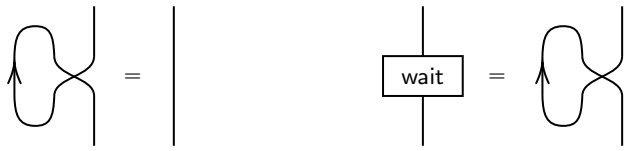


FIGURE 9. The yanking axiom of traced monoidal categories (left) and the morphism wait in feedback monoidal categories (right).

The "state construction", $\mathsf{St}(\bullet)$, realizes the *free* feedback monoidal category. As it happens with feedback monoidal categories, this construction has appeared in the literature in slightly different forms. It has been used for describing a "memoryful geometry of interaction" [HMH14], "stateful resource calculi" [BHP$^+$19], and "processes with feedback" [KSW97, KSW02]. The idea in all of these cases is the same: we allow the morphisms of a monoidal category to depend on a "state space" S, possibly guarded by a functor. Adding a state space is equivalent to freely adding feedback [KSW02, DLGR$^+$21].

Definition 5.4. A *stateful morphism* is a pair (S, f) consisting of a "state space" $S \in \mathsf{C}$ and a morphism $f\colon \mathbf{F}S \otimes X \to S \otimes Y$. We say that two stateful morphisms are *sliding equivalent* if they are related by the smallest equivalence relation satisfying $(S, (\mathbf{F}r \otimes \mathrm{id}) \fatsemi h) \sim (T, h \fatsemi (r \otimes \mathrm{id}))$ for each $h\colon X \otimes \mathbf{F}T \to S \otimes Y$ and each $r\colon S \to T$.

In other words, sliding equivalence is *dinaturality in* S.

Definition 5.5 (St$(\bullet)$ construction, [KSW02, DLGR$^+$21])**.** We write $\mathsf{St}_{\mathbf{F}}(\mathsf{C})$ for the symmetric monoidal category that has the same objects as C and whose morphisms from X to Y are stateful morphisms $f\colon \mathbf{F}S \otimes X \to S \otimes Y$ *up to sliding.*

$$\hom_{\mathsf{St}_{\mathbf{F}}(\mathsf{C})}(X, Y) \coloneqq \int^{S \in \mathsf{C}} \hom_{\mathsf{C}}(\mathbf{F}S \otimes X, S \otimes Y).$$

Theorem 5.6 (see [KSW02])**.** $\mathsf{St}_{\mathbf{F}}(\mathsf{C})$ *is the free feedback monoidal category over* $(\mathsf{C}, \mathbf{F})$.

Proof sketch. Let $(\mathsf{D}, \mathbf{F}^{\mathsf{D}}, \mathbf{fbk}^{\mathsf{D}})$ be any other symmetric monoidal category with an endofunctor, and let $H\colon \mathsf{C} \to \mathsf{D}$ be such that $\mathbf{F} \fatsemi H = H \fatsemi \mathbf{F}^{\mathsf{D}}$. We will prove that it can be extended uniquely to a feedback functor $\tilde{H}\colon \mathsf{St}_{\mathbf{F}}(\mathsf{C}) \to \mathsf{D}$.

It can be proven that any expression involving feedback can be reduced applying the feedback axioms to an expression of the form $\mathbf{fbk}(f)$ for some $f\colon \mathbf{F}S\otimes X \to S\otimes Y$. After this, the definition of $\tilde{H}$ in this morphism is forced to be $\tilde{H}(\mathbf{fbk}\, f) = \mathbf{fbk}^{\mathsf{D}}(\mathsf{D})$. This reduction is uniquely determined, up to sliding, and the morphisms of the $\mathsf{St}(\bullet)$ construction are precisely morphisms $f\colon \mathbf{F}S \otimes X \to S \otimes Y$ quotiented by sliding equivalence. This is the core of the proof by Katis, Sabadini, and Walters [KSW02]. □

5.2. **Dinatural monoidal streams.** Monoidal streams should be, in some sense, the minimal way of adding feedback to a theory of processes. The output of this feedback, however, should be delayed by one unit of time: the category $[\mathbb{N}, \mathsf{C}]$ is naturally equipped with a *delay* endofunctor that shifts a sequence by one. Dinatural monoidal streams form the free delayed-feedback category.

Definition 5.7 (Delay functor)**.** Let $\partial\colon [\mathbb{N}, \mathsf{C}] \to [\mathbb{N}, \mathsf{C}]$ be the endofunctor defined on objects $\mathbb{X} = (X_0, X_1, \dots)$, as $\partial(\mathbb{X}) = (I, X_0, X_1, \dots)$; and on morphisms $\mathbb{f} = (f_0, f_1, \dots)$ as $\partial(\mathbb{f}) = (\mathsf{id}_I, f_0, f_1, \dots)$.

Definition 5.8. The set of *dinatural monoidal streams* from $\mathbb{X}$ to $\mathbb{Y}$ is $\mathbf{R}(\mathbb{X}, \mathbb{Y})$, the hom-set of the free feedback monoidal category over $([\mathbb{N}, \mathsf{C}], \partial)$.

We characterize now dinatural streams in terms of dinatural sequences and the $\mathsf{St}(\bullet)$-construction.

Theorem 5.9. *Dinatural sequences are the explicit construction of dinatural streams,* $\mathbf{R}(\mathbb{X}, \mathbb{Y}) \cong \mathsf{St}_{\partial}([\mathbb{N}, \mathsf{C}])(\mathbb{X}, \mathbb{Y})$*, naturally on* $\mathbb{X}$ *and* $\mathbb{Y}$*.*

Proof. Dinatural sequences coincide, by Definition 5.5, with the state construction, $\mathbf{Dnt}(\mathbb{X}, \mathbb{Y}) = \mathsf{St}_{\partial}([\mathbb{N}, \mathsf{C}])(\mathbb{X}, \mathbb{Y})$. This is still different from our definition of dinatural monoidal streams as morphisms of the free feedback monoidal category; but it follows by Theorem 5.6 that they both categories are equivalent, inducing a natural bijection on the hom-sets. □

As a consequence, the calculus of signal flow graphs given by the syntax of feedback monoidal categories is sound and complete for dinatural equivalence over $[\mathbb{N}, \mathsf{C}]$.

5.3. **Towards observational processes.** Dinatural sequences were an improvement over intensional sequences because they allowed us to equate process descriptions that were *essentially the same*. However, we could still have two processes that are "observationally the same" without them being described in the same way.

Remark 5.10 (Followed by)**.** As we saw in the Introduction, *"followed by"* is a crucial operation in dataflow programming. Any sequence can be decomposed as $\mathbb{X} \cong X_0 \cdot \partial(\mathbb{X}^+)$.[1] We call *"followed by"* the coherence isomorphism in $[\mathbb{N}, \mathsf{C}]$ that witnesses this decomposition.

$$\mathsf{fby}_{\mathbb{X}}\colon X_0 \cdot \partial(\mathbb{X}^+) \to \mathbb{X}$$

In the case of constant sequences $\mathbb{X} = (X, X, \dots)$, we have that $\mathbb{X}^+ = \mathbb{X}$; which means that "followed by" has type $\mathsf{fby}_{\mathbb{X}}\colon X \cdot \partial\mathbb{X} \to \mathbb{X}$.

[1] This can also be seen as the isomorphism making "sequences" a final coalgebra. That is, the first slogan we saw in Section 3.1.

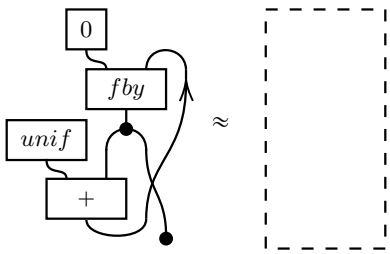


FIGURE 10. *Observationally*, a silent process does nothing.

Example 5.11. Consider the dinatural stateful sequence, in any cartesian monoidal category, that saves the first input to memory without ever producing an output. *Observationally*, this is no different from simply discarding the first input, $(!)_X : X \to 1$. However, in principle, we cannot show that these are *dinaturally equal*, that is, $\mathbf{fbk}(\mathsf{fby}_{\mathbb{X}}) \neq (!)_X$.

More generally, discarding the result of any stochastic or deterministic signal flow graph is, *observationally*, the same as doing nothing (Figure 10, consequence of Proposition 6.9).

6. Observational Monoidal Streams

In this section, we introduce our definitive *monoidal streams*: *observational streams* (Definition 6.2). Their explicit construction is given by observational sequences: dinatural sequences quotiented by observational equivalence.

Intuitively, two processes are observationally equal if they are "equal up to stage n", for any $n \in \mathbb{N}$. We show that, in sufficiently well-behaved monoidal categories (which we call *productive*, Definition 6.11), the set of observational sequences given some inputs and outputs is the final coalgebra of a fixpoint equation (Figure 11). The name "observational equivalence" is commonly used to denote equality on the final coalgebra: Theorem 6.19 justifies our use of the term.

6.1. Observational streams.

We saw in Section 3 that we can define intensional sequences as a solution to a fixpoint equation. We now consider the same equation, just substituting the coproduct for a coend.

Definition 6.1. For a monoidal category C, the endofunctor $(\hom \odot \bullet) \colon [[\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}], \mathsf{Set}] \to [[\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}], \mathsf{Set}]$ is defined by

$$(\hom \odot Q)(\mathbb{X}; \mathbb{Y}) \coloneqq \int^{M \in \mathsf{C}} \hom(X_0, M \otimes Y_0) \times Q(M \cdot \mathbb{X}^+, \mathbb{Y}^+).$$

Definition 6.2 (Observational streams)**.** The set of *observational monoidal streams*, depending on inputs and outputs, is the functor $\mathbf{Q} \colon [\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}] \to \mathsf{Set}$ given by the final fixpoint of the equation in Figure 11.

$$\mathbf{Q}(\mathbb{X}, \mathbb{Y}) \cong \int^{M \in \mathsf{C}} \hom(X_0, M \otimes Y_0) \times \mathbf{Q}(M \cdot \mathbb{X}^+, \mathbb{Y}^+).$$

FIGURE 11. Fixpoint equation for observational streams.

The explicit construction of this final fixpoint will be given by observational sequences (Theorem 6.19).

6.2. **Observational sequences.** We said that observational equivalence is "equality up to stage n", so our first step will be to define what it means to truncate an dinatural sequence at any given $n \in \mathbb{N}$.

Definition 6.3 (n-Stage process)**.** An *n-stage process* from inputs $\mathbb{X} = (X_0, X_1, \ldots)$ to outputs $\mathbb{Y} = (Y_0, Y_1, \ldots)$ is an element of the set

$$\mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}) = \int^{M_0,\ldots,M_n} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, M_i \otimes Y_i).$$

Remark 6.4. In other words, n-stage processes are n-tuples $(f_i \colon M_{i-1} \otimes X_i \to M_i \otimes Y_i)_{i=0}^n$, for some choice of M_i *up to dinaturality*, that we write as

$$\langle f_0 | f_1 | \ldots | f_n | \in \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}).$$

In this notation, *dinaturality* means that morphisms can *slide past the bars*. That is, for any $r_i \colon N_i \to M_i$ and any tuple, dinaturality says that

$$\langle f_0 \fatsemi (r_0 \otimes \mathrm{id}) | f_1 \fatsemi (r_1 \otimes \mathrm{id}) | \ldots | f_n \fatsemi (r_n \otimes \mathrm{id}) | = \langle f_0 | (r_0 \otimes \mathrm{id}) \fatsemi f_1 | \ldots | (r_{n-1} \otimes \mathrm{id}) \fatsemi f_n | .$$

Note that the last r_n is removed by dinaturality.

Definition 6.5 (Truncation)**.** Let $\langle f_n \colon M_{n-1} \otimes X_n \to M_n \otimes Y_n \rangle \in \mathbf{Dnt}(\mathbb{X}, \mathbb{Y})$ be an dinatural sequence. Its *kth-truncation* is the element $\langle f_0 | \ldots | f_k | \in \mathbf{Stage}_k(\mathbb{X}, \mathbb{Y})$. Truncation is well-defined under dinatural equivalence (see Remark 6.4).

For $k \leqslant n$, the *kth-truncation* of an n-stage process given by $\langle f_0 | f_1 | \ldots | f_n | \in \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y})$ is $\langle f_0 | \ldots | f_k | \in \mathbf{Stage}_k(\mathbb{X}, \mathbb{Y})$. This induces functions $\pi_{n,k} \colon \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}) \to \mathbf{Stage}_k(\mathbb{X}, \mathbb{Y})$, with the property that $\pi_{n,m} \fatsemi \pi_{m,k} = \pi_{n,k}$.

Definition 6.6 (Observational equivalence)**.** Two dinatural stateful sequences

$$\langle f \rangle_{n \in \mathbb{N}}, \langle g \rangle_{n \in \mathbb{N}} \in \int^{M \in [\mathbb{N}, \mathsf{C}]} \prod_{i=0}^{\infty} \hom(M_{i-1} \otimes X_i, Y_i \otimes M_i)$$

are *observationally equivalent* when all their n-stage truncations are equal. That is, $\langle f_0 | \ldots | f_n | = \langle g_0 | \ldots | g_n |$, for each $n \in \mathbb{N}$. We write this as $f \approx g$.

Remark 6.7. Formally, this is to say that the sequences $\langle f \rangle_{n \in \mathbb{N}}$ and $\langle g \rangle_{n \in \mathbb{N}}$ have the same image on the limit

$$\lim_n \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}),$$

over the chain $\pi_{n,k} \colon \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}) \to \mathbf{Stage}_k(\mathbb{X}, \mathbb{Y})$.

Definition 6.8. An *observational sequence* from $\mathbb{X}$ to $\mathbb{Y}$ is an equivalence class

$$[\langle f_n \colon M_{n-1} \otimes X_n \to M_n \otimes Y_n \rangle_{n \in \mathbb{N}}]_{\approx}$$

of dinatural sequences under observational equivalence. In other words, the set of observational sequences is

$$\mathbf{Obs}(\mathbb{X}, \mathbb{Y}) \cong \left(\int^{M \in [\mathbb{N}, \mathsf{C}]} \prod_{i=0}^{\infty} \hom(M_{i-1} \otimes X_i, M_i \otimes Y_i) \right) \Big/ \approx .$$

Discarding the output of a process is, *observationally*, the same as discarding its input (cf. Figure 10).

Proposition 6.9. *In any semicartesian category (any symmetric monoidal category whose unit is terminal), observational sequences with no output coincide with the discard sequence.*

Proof. Let $\downarrow_X$ indicate the unique morphism to the terminal object in the base category C. For a stream of objects $\mathbb{X}$, the discard sequence is $\langle \downarrow_{X_n} \rangle_{n\in\mathbb{N}}$. Consider an observational sequence $\langle f_n \rangle_{n\in\mathbb{N}} \colon \mathbb{X} \to \mathbb{I}$ with no output. We show by induction that $\langle f_0 | \cdots | f_n \fatsemi \downarrow_{M_n} | = \langle \downarrow_{X_0} | \cdots | \downarrow_{X_n} |$ without sliding on the last memory M_n. For $n = 0$, $f_0 \fatsemi \downarrow_{M_0} = \downarrow_{X_0}$ by semicartesianity in C, then $\langle f_0 \fatsemi \downarrow_{M_0} | = \langle \downarrow_{X_0} |$. Suppose that $\langle f_0 | \cdots | f_n \fatsemi \downarrow_{M_n} | = \langle \downarrow_{X_0} | \cdots | \downarrow_{X_n} |$ without sliding on the last memory M_n, then,

$$
\begin{aligned}
& \langle f_0 | \cdots | f_n | f_{n+1} \fatsemi \downarrow_{M_{n+1}} | \\
= \; & \langle f_0 | \cdots | f_n | \downarrow_{M_n} \otimes \downarrow_{X_{n+1}} | && \text{(by semicartesianity)} \\
= \; & \langle f_0 | \cdots | f_n \fatsemi \downarrow_{M_n} | \downarrow_{X_{n+1}} | && \text{(by dinaturality)} \\
= \; & \langle \downarrow_{X_0} | \cdots | \downarrow_{X_n} | \downarrow_{X_{n+1}} | && \text{(by induction).}
\end{aligned}
$$

Thus, we can rewrite the truncations of $\langle f_n \rangle_{n\in\mathbb{N}}$, by dinaturality, as follows

$$\langle f_0 | \cdots | f_n | = \langle f_0 | \cdots | f_n \fatsemi \downarrow_{M_n} | = \langle \downarrow_{X_0} | \cdots | \downarrow_{X_n} | \,.$$

This proves that $\langle f_n \rangle_{n\in\mathbb{N}} = \langle \downarrow_{X_n} \rangle_{n\in\mathbb{N}}$ as observational sequences. □

6.3. **Productive categories.** The interaction between dinatural and observational equivalence is of particular interest in some well-behaved categories that we call *productive categories*. In productive categories, observational sequences are the final fixpoint of an equation (Theorem 6.19), analogous to that of Section 3.

An important property of programs is *termination*: a terminating program always halts in a finite amount of time. However, some programs (such as servers, drivers) are not actually intended to terminate but to produce infinite output streams. A more appropriate notion in these cases is that of *productivity*: a program that outputs an infinite stream of data is productive if each individual component of the stream is produced in finite time. To quip, *"a productive stream is a terminating first component followed by a productive stream"*.

The first component of our streams is only defined *up to some future*. It is an equivalence class $\alpha \in \mathbf{Stage}_1(\mathbb{X}, \mathbb{Y})$, with representatives $\alpha_i \colon X_0 \to M_i \otimes Y_0$ that do not necessarily share the same memory. But, if it does terminate, there is a process $\alpha_0 \colon X_0 \to M_0 \otimes Y_0$ in our theory representing the process just until Y_0 is output.

Definition 6.10 (Terminating component)**.** A single stage (0-stage) process $\alpha \in \mathbf{Stage}_0(\mathbb{X}, \mathbb{Y})$ is *terminating relative to* C if there exist M_0 and $\alpha_0 \colon X_0 \to M_0 \otimes Y_0$ such that each one of its representatives, $\langle \alpha' | = \alpha$, can be written as $\alpha' = \alpha_0 \fatsemi (s' \otimes \mathrm{id})$ for some $s' \colon M_0 \to M^{(i)}$.

The morphisms (e.g. s' or s'') represent what is unique to each representative (e.g. α', or α'', respectively), and so we ask that, for any $u \colon M_0 \otimes A \to U \otimes B$ and $v \colon M_0 \otimes A \to V \otimes B$, the equality $\langle \alpha' \otimes \mathrm{id}_A \fatsemi u \otimes \mathrm{id}_{Y_0} | = \langle \alpha'' \otimes \mathrm{id}_A \fatsemi v \otimes \mathrm{id}_{Y_0} |$ implies $\langle s' \otimes \mathrm{id}_A \fatsemi u | = \langle s'' \otimes \mathrm{id}_A \fatsemi v |$.

Definition 6.11 (Productive category)**.** A symmetric monoidal category $(\mathsf{C}, \otimes, I)$ is *productive* when every 0-stage process is terminating relative to C.

Remark 6.12. Cartesian monoidal categories are productive (Proposition 8.7). *Markov categories* [Fri20] with *conditionals* and *ranges* are productive (Theorem 9.11). Free symmetric monoidal categories and compact closed categories are always productive.

6.4. **Observational streams in productive categories.** We conclude this section proving the following Theorem 6.19: whenever the category is productive, observational sequences are explicit construction of observational streams. The proof shows that, from the mild assumption of producitvity, we can extract all of the relevant mathematical structure to make the fixpoint equation work as expected.

Definition 6.13 (Truncating coherently)**.** Let $f_n^k \colon M_{n-1} \otimes X_n \to M_n \otimes Y_n$ be a family of families of morphisms of increasing length, indexed by $k \in \mathbb{N}$ and $n \leqslant k$. We say that this family *truncates coherently* if $\langle f_0^p | \dots | f_n^p | = \langle f_0^q | \dots | f_n^q |$ for each $p, q \in \mathbb{N}$ and each $n \leqslant \min\{p, q\}$.

Lemma 6.14 (Factoring a family of processes)**.** *In a productive category, let $(\langle f_0^k | \dots | f_k^k |)_{k\in\mathbb{N}}$ be a sequence of sequences that truncates coherently. Then, there exists another sequence $h_i \colon N_{i-1} \otimes X_i \to N_i \otimes Y_i$, together with morphisms $s_i^k \colon N_i \to M_i$ satisfying $s_{i-1}^k f_i^k = h_i s_i^k$ and such that, for each $k \in \mathbb{N}$ and each $n \leqslant k$, $\langle f_0^k | \dots | f_n^k | = \langle h_0 | \dots | h_n |$. Moreover, this family h_i is such that $\langle h_0 \dots h_n s_n^p u | = \langle h_0 \dots h_n s_n^q v |$ implies $\langle s_n^p u | = \langle s_n^q v |$.*

Proof. We construct the family by induction. In the case $n = 0$, we use that the family truncates coherently to have that $\langle f_0^p | = \langle f_0^q |$ and thus, by productivity, create an h_0 with $f_0^k = h_0 s_0^k$ such that $\langle h_0 s_0^p u | = \langle h_0 s_0^q v |$ implies $\langle s_0^p u | = \langle s_0^q v |$.

In the general case, assume we already have constructed $h_0, \dots, h_{n-1}$ with $s_{i-1}^k f_i^k = h_i s_i^k$ such that, for each $k \in \mathbb{N}$ and $\langle f_0^k | \dots | f_{n-1}^k | = \langle h_0 | \dots | h_{n-1} |$. Moreover, $\langle h_0 \dots h_{n-1} s_{n-1}^p u | = \langle h_0 \dots h_{n-1} s_{n-1}^q v |$ implies $\langle s_{n-1}^p u | = \langle s_{n-1}^q v |$.

In this case, we use the fact that composition "along a bar" is dinatural: $\langle f_0^p | \dots | f_n^p | = \langle f_0^q | \dots | f_n^q |$ implies that $\langle \underline{f_0^p \dots f_n^p} | = \langle \underline{f_0^q \dots f_n^q} |$. This can be then rewritten as

$$\langle h_0 \dots h_{n-1} s_{n-1}^p f_n^p | = \langle h_0 \dots h_{n-1} s_{n-1}^q f_n^q | \,,$$

which in turn implies $\langle s_{n-1}^p f_n^p | = \langle s_{n-1}^q f_n^q |$. By productivity, there exists h_n with $s_{n-1}^k f_n^k = h_n s_n^k$ such that $\langle f_0^k | \dots | f_n^k | = \langle h_0 | \dots | h_n |$.

Finally, assume that $\langle h_0 \dots h_{n-1} h_n s_n^p u | = \langle h_0 \dots h_{n-1} h_n s_n^q v |$. Thus, we have

$$\langle h_0 \dots h_{n-1} s_{n-1}^p f_n^p u | = \langle h_0 \dots h_{n-1} s_{n-1}^q f_n^q v |$$

and $\langle s_{n-1}^p f_n^p u | = \langle s_{n-1}^q f_n^q v |$. This can be rewritten as $\langle h_n s_n^p u | = \langle h_n s_n^q v |$, which in turn implies $\langle s_n^p u | = \langle s_n^q v |$. We have shown that the h_n that we constructed satisfies the desired property. □

Lemma 6.15. *In a productive category, the set of observational sequences is isomorphic to the limit of the terminal sequence of the endofunctor* $(\hom \odot \bullet)$ *via the canonical map between them.*

Proof. We start by noting that observational equivalence of sequences is, by definition, the same thing as being equal under the canonical map to the limit of the terminal sequence

$$\lim_n \int^{M_0,\dots,M_n} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, M_i \otimes Y_i).$$

We will show that this canonical map is surjective. This means that the domain quotiented by equality under the map is isomorphic to the codomain, q.e.d.

Indeed, given any family f_n^k that truncates coherently, we can apply Lemma 6.14 to find a sequence h_i such that $\langle f_0^k | \dots | f_n^k | = \langle h_0 | \dots | h_n |$. This means that it is the image of the stateful sequence h_i. □

Lemma 6.16 (Factoring two processes)**.** *In a productive category, let* $\langle f_0| = \langle g_0|$*. Then there exists* h_0 *with* $f_0 = h_0 s_0$ *and* $g_0 = h_0 t_0$ *such that*

$$\langle f_0| \dots |f_n| = \langle g_0| \dots |g_n|$$

implies the existence of a family h_i *together with* s_i *and* t_i *such that* $s_{i-1} f_i = h_i s_i$ *and* $t_{i-1} g_i = h_i t_i$*; and moreover, such that*

$$\langle h_0 \dots h_n s_n u| = \langle h_0 \dots h_n t_n v| \text{ implies } \langle s_n u| = \langle t_n v| .$$

Proof. By productivity, we can find such a factorization $f_0 = h_0 s_0$ and $g_0 = h_0 t_0$.

Assume now that we have a family of morphisms such that $\langle f_0| \dots |f_n| = \langle g_0| \dots |g_n|$. We proceed by induction on n, the size of the family. The case $n = 0$ follows from the definition of productive category.

In the general case, we will construct the relevant h_n. The assumption $\langle f_0| \dots |f_n| = \langle g_0| \dots |g_n|$ implies, in particular, that $\langle f_0| \dots |f_{n-1}| = \langle g_0| \dots |g_{n-1}|$. Thus, by induction hypothesis, there exist $h_1, \dots, h_{n-1}$ together with $s_{i-1} f_i = h_i s_i$ and $t_{i-1} g_i = h_i t_i$, such that

$$\langle h_0 \dots h_n s_{n-1} u| = \langle h_0 \dots h_n t_{n-1} v| \text{ implies } \langle s_{n-1} u| = \langle t_{n-1} v| .$$

We know that $\langle f_0 \dots f_n| = \langle g_0 \dots g_n|$ and thus,

$$\langle h_0 \dots h_{n-1} s_{n-1} f_n| = \langle h_0 \dots h_{n-1} t_{n-1} g_n| ,$$

which, by induction hypothesis, implies $\langle s_{n-1} f_n| = \langle t_{n-1} g_n|$. By productivity, there exists h_n with $s_{n-1} f_n = h_n s_n$ and $t_{n-1} g_n = h_n t_n$ such that $\langle h_n s_n u| = \langle h_n t_n v|$ implies $\langle s_n u| = \langle t_n v|$.

Finally, assume that $\langle h_0 \dots h_{n-1} h_n s_n u| = \langle h_0 \dots h_{n-1} h_n t_n v|$. Thus, we have

$$\langle h_0 \dots h_{n-1} s_{n-1} f_n u| = \langle h_0 \dots h_{n-1} t_{n-1} g_n v|$$

and $\langle s_{n-1} f_n u| = \langle t_{n-1} g_n v|$. This can be rewritten as $\langle h_n s_n u| = \langle h_n t_n v|$, which in turn implies $\langle s_n u| = \langle t_n v|$. We have shown that the h_n that we constructed satisfies the desired property. □

Lemma 6.17 (Removing the first step)**.** *In a productive category, let* $\langle f_0| = \langle g_0|$*. Then there exists* h *with* $f_0 = hs$ *and* $g_0 = ht$ *such that* $\langle f_0| \dots |f_n| = \langle g_0| \dots |g_n|$ *implies* $\langle s f_1| \dots |f_n| = \langle t g_1| \dots |g_n|$*.*

Proof. By Lemma 6.16, we obtain a factorization $f_0 = hs$ and $g_0 = ht$. Moreover, each time that we have $\langle f_0| \dots |f_n| = \langle g_0| \dots |g_n|$, we can obtain a family h_i together with s_i and t_i such that $s_{i-1} f_i = h_i s_i$ and $t_{i-1} f_i = h_i t_i$. Using the fact that $\langle h_n s_n| = \langle h_n t_n|$, we have that $\langle h_1| \dots |h_n s_n| = \langle h_1| \dots |h_n t_n|$, which can be rewritten using dinaturality as $\langle s_0 f_1| \dots |f_n| = \langle t_0 g_1| \dots |g_n|$. □

Lemma 6.18. *In a productive category, the final coalgebra of the endofunctor* $(\hom \odot \bullet)$ *does exist and it is given by the limit of the terminal sequence*

$$L \coloneqq \lim_n \int^{M_0, \dots, M_n} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, M_i \otimes Y_i).$$

Proof. We will apply Theorem 2.5. The endofunctor $(\hom \odot \bullet)$ acts on the category $[([\mathbb{N}, \mathsf{C}])^{op} \times [\mathbb{N}, \mathsf{C}], \mathsf{Set}]$, which, being a presheaf category, has all small limits. We will show that there is an isomorphism $\hom \odot L \cong L$ given by the canonical morphism between them.

First, note that the set $L(\mathbb{X};\mathbb{Y})$ is, explicitly,

$$\lim_n \int^{M_1,\dots,M_n} \prod_{i=1}^{n} \hom(M_{i-1}\otimes X_i, M_i\otimes Y_i).$$

A generic element from this set is a *sequence of sequences of increasing length*. Moreover, the sequences must *truncate coherently* (Definition 6.13).

Secondly, note that the set $(\hom \odot L)(\mathbb{X};\mathbb{Y})$ is, explicitly,

$$\int^{M_0} \left(\hom(X_0, M_0\otimes Y_0) \times \lim_n \int^{M_1,\dots,M_n} \prod_{i=1}^{n} \hom(M_{i-1}\otimes X_i, M_i\otimes Y_i) \right).$$

A generic element from this set is of the form

$$\langle f | (\langle f_1^k | \dots | f_k^k |)_{k\in\mathbb{N}} | \,,$$

that is, a pair consisting on a first morphism $f\colon X_0 \to Y_0 \otimes M_0$ and a family of sequences $(\langle f_1^k | \dots | f_k^k |)$, quotiented by dinaturality of M_0 and truncating coherently. The canonical map to $L(\mathbb{X};\mathbb{Y})$ maps this generic element to the family of sequences $(\langle f_0 | f_1^k | \dots | f_k^k |)_{k\in\mathbb{N}}$, which truncates coherently because the previous family did and we are precomposing with f_0, which is dinatural.

Thirdly, this map is injective. Imagine a pair of elements $\langle f_0 | (\langle f_1^k | \dots | f_k^k |)_{k\in\mathbb{N}} |$ and $\langle g_0 | (\langle g_1^k | \dots | g_k^k |)_{k\in\mathbb{N}} |$ that have the same image, meaning that, for each $k \in \mathbb{N}$,

$$\langle f_0 | f_1^k | \dots | f_k^k | = \langle g_0 | g_1^k | \dots | g_k^k | \,.$$

By Lemma 6.17, we can find h with $f_0 = hs$ and $g_0 = ht$ such that, for each $k \in \mathbb{N}$, $\langle sf_1^k | \dots | f_k^k | = \langle tg_1^k | \dots | g_k^k |$. Thus,

$$\langle f_0 | (\langle f_1^k | \dots | f_k^k |)_{k\in\mathbb{N}} | = \langle h | (\langle sf_1^k | \dots | f_k^k |)_{k\in\mathbb{N}} | =$$
$$\langle h | (\langle tg_1^k | \dots | g_k^k |)_{k\in\mathbb{N}} | = \langle g_0 | (\langle tg_1^k | \dots | g_k^k |)_{k\in\mathbb{N}} | \,.$$

Finally, this map is also surjective. From Lemma 6.14, it follows that any family that truncates coherently can be equivalently written as $\langle h_0 | \dots | h_n |_{n\in\mathbb{N}}$, which is the image of the element $\langle h_0 | \langle h_1 | \dots | h_n |_{n\in\mathbb{N}}$. □

Theorem 6.19. *Observational sequences are the explicit construction of observational streams when the category is productive. More precisely, in a productive category, the final fixpoint of the equation in Figure 11 is given by the set of observational sequences,* **Obs**.

In a productive category, the final coalgebra of the endofunctor $(\hom \odot \bullet)$ *exists and it is given by the set of stateful sequences quotiented by observational equivalence.*

$$\left(\int^{M\in[\mathbb{N},\mathsf{C}]} \prod_{i=0}^{\infty} \hom(M_{i-1}\otimes X_i, M_i\otimes Y_i) \right) \Big/ \approx$$

Proof sketch. The terminal sequence for this final coalgebra is given by $\mathbf{Stage}_n(\mathbb{X},\mathbb{Y})$. In productive categories, we have proven that the limit $\lim_n \mathbf{Stage}_n(\mathbb{X},\mathbb{Y})$ is a fixpoint of the equation in Figure 11 (Lemma 6.18). The hypothesis of productivity is used to make a particular coend commute with a limit, so that Adamek's theorem can be applied (Theorem 2.5). By Lemma 6.18, we know that the final coalgebra exists and is given by the limit of the terminal sequence. By Lemma 6.15, we know that it is isomorphic to the set of stateful sequences quotiented by observational equivalence. □

7. The Category of Monoidal Streams

We are ready to construct Stream: the feedback monoidal category of monoidal streams. Let us recast the definitive notion of monoidal stream (Definition 6.2) coinductively.

Definition 7.1. A *monoidal stream* $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ is a triple consisting of

- $M(f) \in \mathsf{Obj}(\mathsf{C})$, the *memory*,
- $\mathsf{now}(f) \in \hom(X_0, M(f) \otimes Y_0)$, the *first action*,
- $\mathsf{later}(f) \in \mathsf{Stream}(M(f) \cdot \mathbb{X}^+, \mathbb{Y}^+)$, the *rest of the action*,

quotiented by dinaturality in $M(f)$. Explicitly, monoidal streams are quotiented by the equivalence relation $f \sim g$ generated by

- the existence of $r \colon M(g) \to M(f)$,
- such that $\mathsf{now}(f) = \mathsf{now}(g) \fatsemi r$,
- and such that $r \cdot \mathsf{later}(f) \sim \mathsf{later}(g)$.

Here, $r \cdot \mathsf{later}(f) \in \mathsf{Stream}(M(g) \cdot \mathbb{X}^+, \mathbb{Y}^+)$ is obtained by precomposition of the first action of $\mathsf{later}(f)$ with r.

Remark 7.2. In terms of string diagrams (Figure 12, left), a monoidal stream is defined by two parts connected by some wires and separated by some space representing the context between them. The first part represents the first action which is a morphism in the base monoidal category; the second part represents the rest of the action, which is again a stream. This is, the two parts of the diagram are in two different categories. However, we can still slide morphisms along the wires connecting both parts; this sliding encodes dinaturality (Figure 12, right).

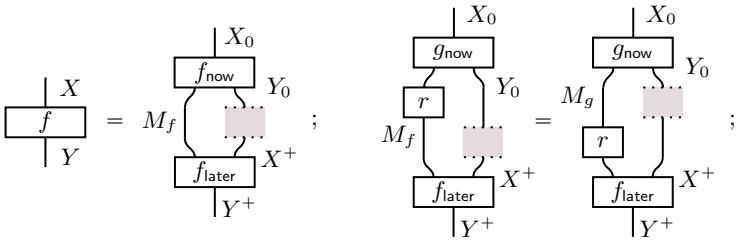


FIGURE 12. String diagrammatic definition of a monoidal stream.

Remark 7.3. This definition of monoidal streams is the coinductive definition of the functor

$$\mathsf{Stream} \colon [\mathbb{N}, \mathsf{C}]^{op} \times [\mathbb{N}, \mathsf{C}] \to \mathsf{Set}.$$

In principle, arbitrary final coalgebras do not need to exist. Moreover, it is usually difficult to explicitly construct such coalgebras [Ada74]. However, in productive categories, this coalgebra does exist and is constructed by observational sequences.

From now on, we reason *coinductively* [KS17], a style particularly suited for all the following definitions. Coinduction is the dual to induction. It uses coalgebra to formalize the reasoning with non-well-founded structures [Par81, Hon81, TR98, AM89], and it has been employed for many decades now for reasoning with infinite data in computer science [Rut00, HJ98]. In practice, we can appeal to the *coinduction hypothesis* after providing the first step. When constructing, we employ the universal property of the final coalgebra to uniquely extend any coalgebra morphism; when proving, we use the fact that two elements of the final coalgebra are equal when they are indistinguishable, or *bisimilar* [BM89, San09].

Coinductive reasoning will not only occur at the level of expressions but at the level of string diagrams. Each string diagram of monoidal streams can be unfold repeatedly using the notation of Figure 12.

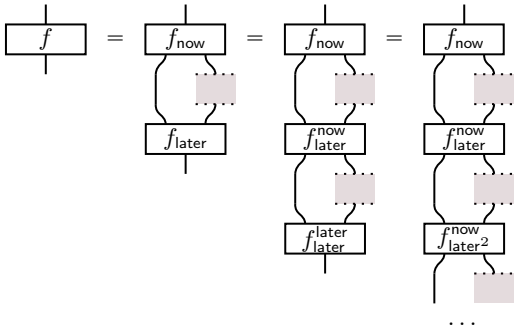


FIGURE 13. Coinductive unfolding of a string diagram.

Moreover, every time we introduce a new morphism, we can define it in terms of a string diagram as in Figure 12: the second part of this string diagram can depend, coinductively, on the same definition we are writing.

7.1. **The Symmetric Monoidal Category of Streams.** The definitions for the operations of sequential and parallel composition are described in two steps. We first define an operation that takes into account an extra *memory channel* (see, for instance, Figure 14); we use this extra generality to strengthen the *coinduction hypothesis*. We then define the desired operation as a particular case of this coinductively defined one.

Definition 7.4 (Sequential composition)**.** Given two streams $f \in \mathsf{Stream}(A \cdot \mathbb{X}, \mathbb{Y})$ and $g \in \mathsf{Stream}(B \cdot \mathbb{Y}, \mathbb{Z})$, we compute $(f^A \fatsemi g^B) \in \mathsf{Stream}((A \otimes B) \cdot \mathbb{X}, \mathbb{Z})$, their *sequential composition with memories A and B*, as

- $M(f^A \fatsemi g^B) = M(f) \otimes M(g)$,
- $\mathsf{now}(f^A \fatsemi g^B) = \sigma \fatsemi (\mathsf{now}(f) \otimes \mathrm{id}) \fatsemi \sigma \fatsemi (\mathsf{now}(g) \otimes \mathrm{id})$,
- $\mathsf{later}(f^A \fatsemi g^B) = \mathsf{later}(f)^{M(f)} \fatsemi \mathsf{later}(g)^{M(g)}$.

We write $(f \fatsemi g)$ for $(f^I \fatsemi g^I) \in \mathsf{Stream}(\mathbb{X}, \mathbb{Z})$; the *sequential composition* of $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ and $g \in \mathsf{Stream}(\mathbb{Y}, \mathbb{Z})$.

Definition 7.5. The *identity* $\mathrm{id}_{\mathbb{X}} \in \mathsf{Stream}(\mathbb{X}, \mathbb{X})$ is defined by $M(\mathrm{id}_{\mathbb{X}}) = I$, $\mathsf{now}(\mathrm{id}_{\mathbb{X}}) = \mathrm{id}_{X_0}$, and $\mathsf{later}(\mathrm{id}_{\mathbb{X}}) = \mathrm{id}_{\mathbb{X}^+}$.

Lemma 7.6. *Sequential composition of streams with memories (Definition 7.4) is associative. Given three streams $f \in \mathsf{Stream}(A \cdot \mathbb{X}, \mathbb{Y})$, $g \in \mathsf{Stream}(B \cdot \mathbb{Y}, \mathbb{Z})$, and $h \in \mathsf{Stream}(B \cdot \mathbb{Z}, \mathbb{W})$; we can compose them in two different ways,*

- $(f^A \fatsemi g^B) \fatsemi h^C \in \mathsf{Stream}((A \otimes B) \otimes C \cdot \mathbb{X}, \mathbb{W})$, *or*
- $f^A \fatsemi (g^B \fatsemi h^C) \in \mathsf{Stream}(A \otimes (B \otimes C) \cdot \mathbb{X}, \mathbb{W})$.

We claim that

$$((f^A \fatsemi g^B) \fatsemi h^C) = \alpha_{A,B,C} \cdot (f^A \fatsemi (g^B \fatsemi h^C)).$$

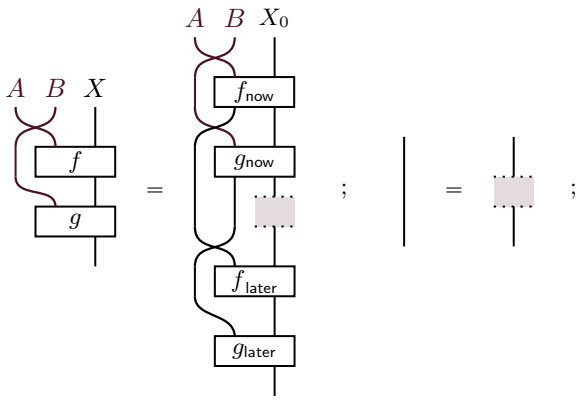


FIGURE 14. Sequential composition and identity of monoidal streams.

Proof. First, we note that both sides of the equation represent streams with different memories.

- $M((f^A \fatsemi g^B) \fatsemi h^C) = (M(f) \otimes M(g)) \otimes M(h)$,
- $M(f^A \fatsemi (g^B \fatsemi h^C)) = M(f) \otimes (M(g) \otimes M(h))$.

We will prove they are related by dinaturality over the associator α. We know that $\mathsf{now}((f^A \fatsemi g^B) \fatsemi h^C) = \mathsf{now}(f^A \fatsemi (g^B \fatsemi h^C))$ by string diagrams (see Figure 15). Then, by coinduction, we know that

$$(\mathsf{later}(f)^{M(f)} \fatsemi \mathsf{later}(g)^{M(g)}) \fatsemi \mathsf{later}(h)^{M(h)} =$$
$$\alpha \cdot (\mathsf{later}(f)^{M(f)} \fatsemi (\mathsf{later}(g)^{M(g)} \fatsemi \mathsf{later}(h)^{M(h)})),$$

that is, $\mathsf{later}((f^A \fatsemi g^B) \fatsemi h^C) = \alpha \cdot \mathsf{later}(f^A \fatsemi (g^B \fatsemi h^C))$. □

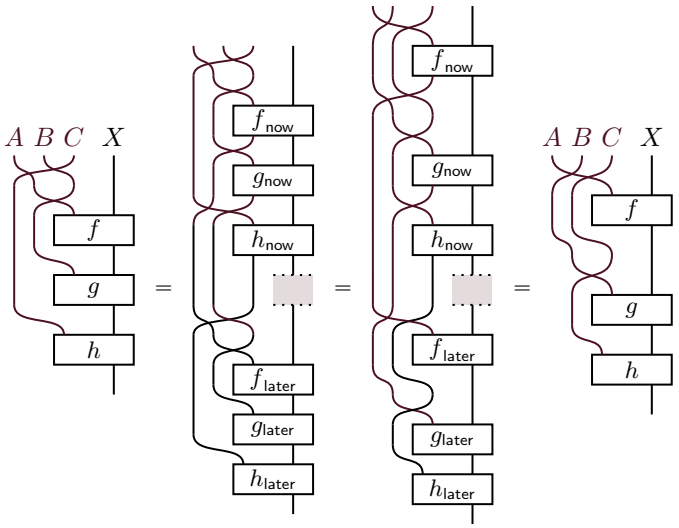


FIGURE 15. Associativity of sequential composition.

Lemma 7.7. *Sequential composition of streams (Definition 7.4) is associative. Given three streams* $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$, $g \in \mathsf{Stream}(\mathbb{Y}, \mathbb{Z})$, *and* $h \in \mathsf{Stream}(\mathbb{Z}, \mathbb{W})$*; we claim that*

$$((f \fatsemi g) \fatsemi h) = (f \fatsemi (g \fatsemi h)).$$

Proof. Direct consequence of Lemma 7.6, after considering the appropriate coherence morphisms. □

Definition 7.8 (Parallel composition)**.** Given two streams $f \in \mathsf{Stream}(A \cdot \mathbb{X}, \mathbb{Y})$ and $g \in \mathsf{Stream}(B \cdot \mathbb{X}', \mathbb{Y}')$, we compute $(f^A \otimes g^B) \in \mathsf{Stream}((A \otimes B) \cdot (\mathbb{X} \otimes \mathbb{X}'), \mathbb{Y} \otimes \mathbb{Y}')$, their *parallel composition with memories* A *and* B, as

- $M(f^A \otimes g^B) = M(f) \otimes M(g)$,
- $\mathsf{now}(f^A \otimes g^B) = \sigma \fatsemi (\mathsf{now}(f) \otimes \mathsf{now}(g)) \fatsemi \sigma$,
- $\mathsf{later}(f^A \otimes g^B) = \mathsf{later}(f)^{M(f)} \otimes \mathsf{later}(g)^{M(g)}$.

We write $(f \otimes g)$ for $(f^I \otimes g^I) \in \mathsf{Stream}(\mathbb{X} \otimes \mathbb{X}', \mathbb{Y} \otimes \mathbb{Y}')$; we call it the *parallel composition* of $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ and $g \in \mathsf{Stream}(\mathbb{X}', \mathbb{Y}')$.

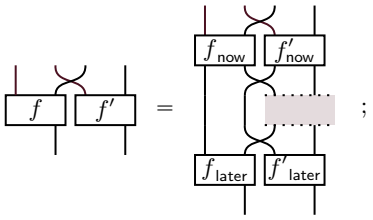


FIGURE 16. Parallel composition.

Lemma 7.9. *Parallel composition of streams with memories is functorial with regards to sequential composition of streams with memories. Given four streams* $f \in \mathsf{Stream}(A \cdot \mathbb{X}, \mathbb{Y})$, $f' \in \mathsf{Stream}(A' \cdot \mathbb{X}', \mathbb{Y}')$, $g \in \mathsf{Stream}(B \cdot \mathbb{Y}, \mathbb{Z})$, *and* $g' \in \mathsf{Stream}(B' \cdot \mathbb{Y}', \mathbb{Z}')$, *we can compose them in two different ways,*

- $(f^A \otimes f'^{A'})^{(A \otimes A')} \fatsemi (g^B \otimes g'^{B'})^{(B \otimes B')}$, *and*
- $(f^A \fatsemi g^B)^{(A \otimes B)} \otimes (f'^{A'} \fatsemi g'^{B'})^{(A' \otimes B')}$,

having slightly different types, respectively,

- $\mathsf{Stream}((A \otimes A') \otimes (B \otimes B') \cdot \mathbb{X} \otimes \mathbb{X}', \mathbb{Z} \otimes \mathbb{Z}')$, *and*
- $\mathsf{Stream}((A \otimes B) \otimes (A' \otimes B') \cdot \mathbb{X} \otimes \mathbb{X}', \mathbb{Z} \otimes \mathbb{Z}')$.

We claim that

$$(f^A \otimes f'^{A'})^{(A \otimes A')} \fatsemi (g^B \otimes g'^{B'})^{(B \otimes B')} = \sigma_{A',B} \cdot (f^A \fatsemi g^B)^{(A \otimes B)} \otimes (f'^{A'} \fatsemi g'^{B'})^{(A' \otimes B')}.$$

Proof. First, we note that both sides of the equation (which, from now on, we call LHS and RHS, respectively) represent strams with different memories.

- $M(LHS) = (M(f) \otimes M(f')) \otimes (M(g) \otimes M(g'))$,
- $M(RHS) = (M(f) \otimes M(g)) \otimes (M(f') \otimes M(g'))$.

We will prove they are related by dinaturality over the symmetry σ. We know that $\mathsf{now}(LHS) \fatsemi \sigma = \mathsf{now}(RHS)$ by string diagrams (see Figure 17). Then, by coinduction, we know that $\mathsf{later}(LHS) = \sigma \cdot \mathsf{later}(RHS)$. □

Lemma 7.10. *Parallel composition of streams is functorial with respect to sequential composition of streams. Given four streams* $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$, $f' \in \mathsf{Stream}(\mathbb{X}', \mathbb{Y}')$, $g \in \mathsf{Stream}(\mathbb{Y}, \mathbb{Z})$, *and* $g' \in \mathsf{Stream}(\mathbb{Y}, \mathbb{Z})$*; we claim that* $(f \otimes f') \fatsemi (g \otimes g') = (f \fatsemi g) \otimes (f' \fatsemi g')$.

Proof. Direct consequence of Lemma 7.9, after considering the appropriate coherence morphisms. □

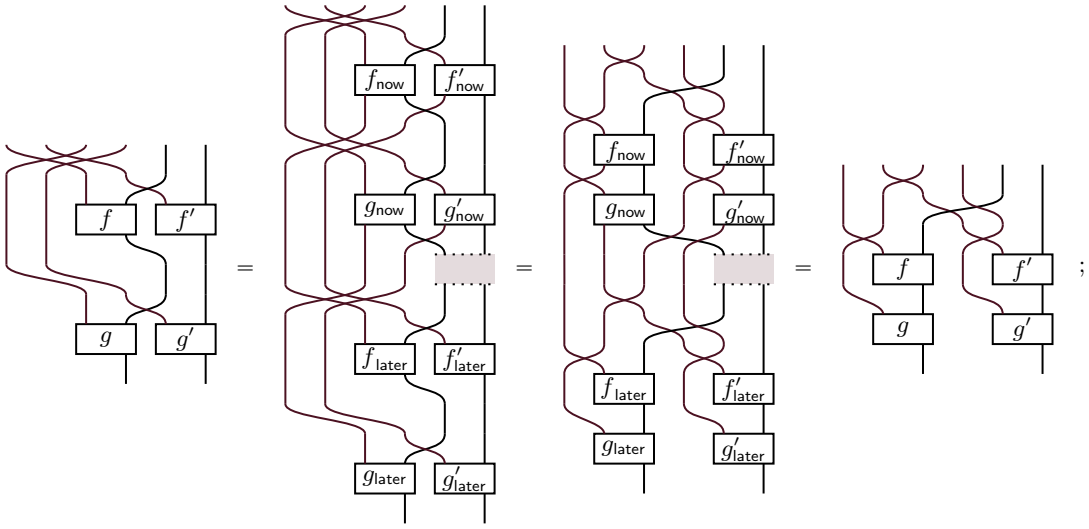


FIGURE 17. Functoriality of parallel composition.

Definition 7.11 (Memoryless and constant streams)**.** Each sequence $\mathbb{f} = (f_0, f_1, \ldots)$, with $f_n \colon X_n \to Y_n$, induces a stream $[\![\mathbb{f}]\!] \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ defined by $M([\![\mathbb{f}]\!]) = I$, $\mathsf{now}([\![\mathbb{f}]\!]) = f_0$, and $\mathsf{later}([\![\mathbb{f}]\!]) = [\![\mathbb{f}^+]\!]$. Streams of this form are called *memoryless*, i.e. their memories are given by the monoidal unit.

Moreover, each morphism $f_0 \colon X \to Y$ induces a *constant* memoryless stream that we call $f \in \mathsf{Stream}(X, Y)$, defined by $M(f) = I$, $\mathsf{now}(f) = f_0$, and $\mathsf{later}(f) = f$.

Theorem 7.12 (see [Rom20])**.** *Monoidal streams over a productive symmetric monoidal category* $(\mathsf{C}, \otimes, I)$ *form a symmetric monoidal category* Stream *with a symmetric monoidal identity-on-objects functor from* $[\mathbb{N}, \mathsf{C}]$.

Proof. Sequential composition of streams (Definition 7.4) is associative (Lemma 7.7) and unital with respect to identities. Parallel composition is bifunctorial with respect to sequential composition (Lemma 7.10); this determines a bifunctor, which is the tensor of the monoidal category. The coherence morphisms and the symmetry can be included from sets, so they still satisfy the pentagon and triangle equations. □

The construction of monoidal streams preserves semicartesianity. This is another way of saying that silent processes are equivalent to doing nothing (Figure 10).

Proposition 7.13. *Monoidal streams over a productive semicartesian category* C *form a semicartesian category.*

Proof. We will show by coinduction that the counit stream is natural, i.e. $f \fatsemi [\!\downarrow_{Y_n}] = \downarrow_M \cdot [\!\downarrow_{X_n}]$ for any stream with memory $f \in \mathsf{Stream}(M \cdot \mathbb{X}, \mathbb{Y})$. We will prove that they are related by dinaturality over the discard map, $\downarrow_{M_0}$. We can see that the first actions are related by semicartesianity,

$$\mathsf{now}(f \fatsemi [\!\downarrow_{Y_n}]) \fatsemi \downarrow_{M_0} = \mathsf{now}(f) \fatsemi (\downarrow_{M_0} \otimes \downarrow_{Y_0}) = \downarrow_M \otimes \downarrow_{X_0} = \mathsf{now}(\downarrow_M \cdot [\!\downarrow_{X_n}]).$$

We can see that the rest of the actions are related by coinduction.

$$\mathsf{later}(f \fatsemi [\!\downarrow_{Y_n}]) = \mathsf{later}(f) \fatsemi [\!\downarrow_{Y_{n+1}}] = \downarrow_{M_0} \cdot [\!\downarrow_{X_{n+1}}] = \downarrow_{M_0} \cdot \mathsf{later}([\!\downarrow_{X_n}]).$$

This shows $f \fatsemi [\downarrow_{Y_n}] = \downarrow_M \cdot [\downarrow_{X_n}]$, by definition of equality in monoidal streams (Definition 7.1). □

7.2. **Delayed feedback for streams.** Monoidal streams form a *delayed feedback* monoidal category. Given some stream in $\mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$, we can create a new stream in $\mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ that passes the output in $\mathbb{S}$ as a memory channel that gets used as the input in $\partial\mathbb{S}$. As a consequence, the category of monoidal streams has a graphical calculus given by that of feedback monoidal categories. This graphical calculus is complete for dinatural equivalence (as we saw in Theorem 5.9).

Definition 7.14 (Delay functor)**.** The functor from Definition 5.7 can be lifted to a monoidal functor $\partial \colon \mathsf{Stream} \to \mathsf{Stream}$ that acts on objects in the same way. It acts on morphisms by sending a stream $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ to the stream given by $M(\partial f) = I$, $\mathsf{now}(\partial f) = \mathrm{id}_I$ and $\mathsf{later}(\partial f) = f$.

Definition 7.15 (Feedback operation)**.** Given any morphism of the form $f \in \mathsf{Stream}(N \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$, we define $\mathbf{fbk}(f^N) \in \mathsf{Stream}(N \cdot \mathbb{X}, \mathbb{Y})$ as

- $M(\mathbf{fbk}(f^N)) = M(f) \otimes S_0$,
- $\mathsf{now}(\mathbf{fbk}(f^N)) = \mathsf{now}(f)$ and
- $\mathsf{later}(\mathbf{fbk}(f^N)) = \mathbf{fbk}(\mathsf{later}(f)^{M(f) \otimes S_0})$.

We write $\mathbf{fbk}(f) \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ for $\mathbf{fbk}(f^I)$, the feedback of $f \in \mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$ tensor.

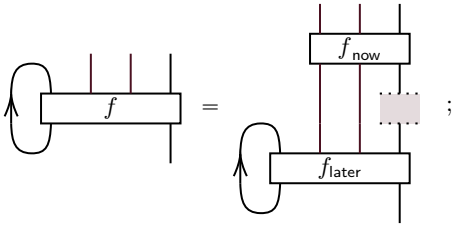


FIGURE 18. Definition of feedback.

Lemma 7.16. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the tightening axiom (A1). Given three streams* $u \in \mathsf{Stream}(A \cdot \mathbb{X}', \mathbb{X})$*,* $f \in \mathsf{Stream}(B \otimes T \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, and* $v \in \mathsf{Stream}(C \cdot \mathbb{Y}, \mathbb{Y}')$*; we claim that*

$$\mathbf{fbk}^S(u^A \fatsemi f^B \fatsemi v^C) = \sigma \cdot u^A \fatsemi \mathbf{fbk}^S(f^{B \otimes T}) \fatsemi v^C.$$

Proof. First, we note that both sides of the equation (which, from now on, we call LHS and RHS, respectively) represent streams with different memories.

- $M(LHS) = A \otimes B \otimes C \otimes T$,
- $M(RHS) = A \otimes B \otimes T \otimes C$.

We will prove that they are related by dinaturality over the symmetry σ. We know that $\mathsf{now}(LHS) \fatsemi \sigma = \mathsf{now}(RHS)$ by string diagrams (see Figure 19). Then, by coinduction, we know that $\mathsf{later}(LHS) = \sigma \cdot \mathsf{later}(RHS)$. □

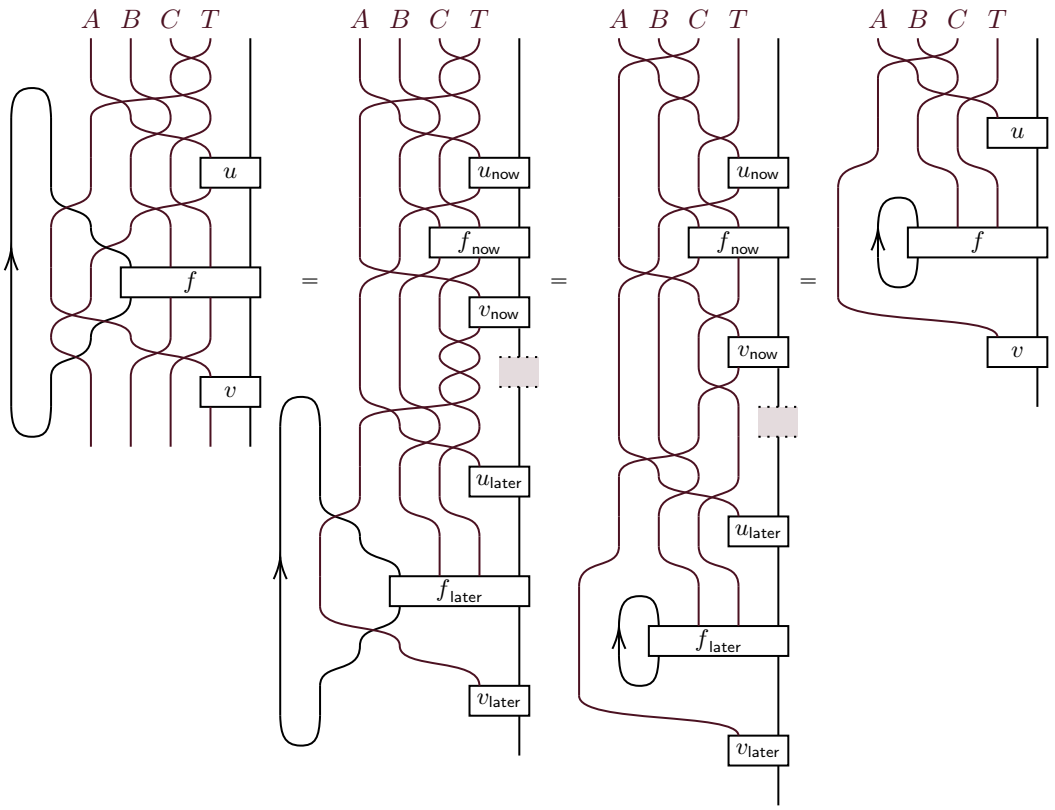


FIGURE 19. The tightening axiom (A1).

Lemma 7.17. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *satisfies the tightening axiom (A1). Given streams* $u \in \mathsf{Stream}(\mathbb{X}', \mathbb{X})$*,* $f \in \mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, and* $v \in \mathsf{Stream}(\mathbb{Y}, \mathbb{Y}')$*; we claim that*

$$\mathbf{fbk}^S(u \fatsemi f \fatsemi v) = u \fatsemi \mathbf{fbk}^S(f) \fatsemi v.$$

Proof. Consequence of Lemma 7.16, after applying the necessary coherence morphisms. □

Lemma 7.18. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the vanishing axiom (A2). Given a stream* $f \in \mathsf{Stream}(A \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, we claim that*

$$\mathbf{fbk}^I(f^A) = \rho \cdot f.$$

Proof. First, we note that both sides of the equation represent streams with different memories, $M(\mathbf{fbk}^I(f^A)) = M(f) \otimes I$. We will prove that they are related by dinaturality over the right unitor ρ. We know that $\mathsf{now}(\mathbf{fbk}^I(f^A)) = \mathsf{now}(f)$ by definition. Then, by coinduction, we konw that $\mathsf{later}(\mathbf{fbk}^I(f^A)) = \rho \cdot \mathsf{later}(f)$. □

Lemma 7.19. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *satisfies the vanishing axiom (A2). Given a stream* $f \in \mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, we claim that*

$$\mathbf{fbk}^I(f) = f.$$

Proof. Consequence of Lemma 7.18, after applying the necessary coherence morphisms. □

Lemma 7.20. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the joining axiom (A3). Given a stream* $f \in \mathsf{Stream}((A \otimes P \otimes Q) \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, we claim that*

$$\mathbf{fbk}^{S \otimes T}(f^{A \otimes (P \otimes Q)}) = \alpha \cdot \mathbf{fbk}^T(\sigma \cdot \mathbf{fbk}^S(\sigma \cdot f^{(A \otimes Q) \otimes P})).$$

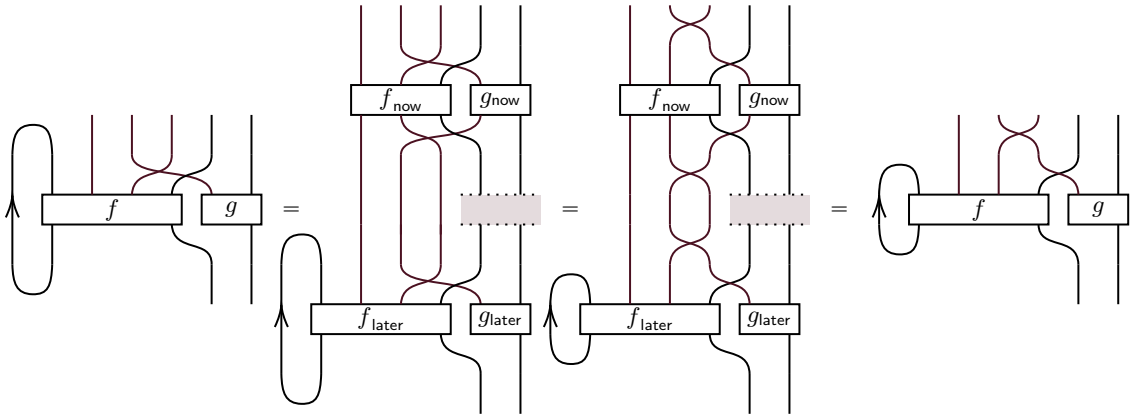


FIGURE 20. The strength axiom (A4).

Proof. First, we note that both sides of the equation (which, from now on, we call LHS and RHS, respectively) represent strams with different memories.

- $M(LHS) = M(f) \otimes (S_0 \otimes T_0)$,
- $M(RHS) = (M(f) \otimes S_0) \otimes T_0$.

We will prove that they are related by dinaturality over the associator α. We know that $\mathsf{now}(LHS) \fatsemi \alpha = \mathsf{now}(RHS)$ by definition. Then, by coinduction, we know that $\mathsf{later}(LHS) = \alpha \cdot \mathsf{later}(RHS)$. □

Lemma 7.21. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *satisfies the joining axiom (A3). Given a stream* $f \in \mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, we claim that*

$$\mathbf{fbk}^{S \otimes T}(f) = \mathbf{fbk}^{T}(\mathbf{fbk}^{S}(f)).$$

Proof. Consequence of Lemma 7.20, after applying the necessary coherence morphisms. □

Lemma 7.22. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the strength axiom (A4). Given two streams* $f \in \mathsf{Stream}((A \otimes P) \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, and* $g \in \mathsf{Stream}(B \cdot \mathbb{X}', \mathbb{Y}')$*, we claim that*

$$\alpha \cdot \mathbf{fbk}^{S}(f^{A} \otimes g^{B}) = \mathbf{fbk}^{S}(f^{A \otimes P})^{A \otimes P} \otimes g^{B}.$$

Proof. First, we note that both sides of the equation (which, from now on, we call LHS and RHS, respectively) represent streams with different memories.

- $M(LHS) = (M(f) \otimes S_0) \otimes M(g)$,
- $M(RHS) = M(f) \otimes (S_0 \otimes M(g))$.

We will prove that they are related by dinaturality over the symmetry α. We know that $\mathsf{now}(LHS) = \mathsf{now}(RHS) \fatsemi \alpha$ by string diagrams (see Figure 20). Then, by coinduction, we know that $\alpha \cdot \mathsf{later}(LHS) = \mathsf{later}(RHS)$. □

Lemma 7.23. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the strength axiom (A4). Given two streams* $f \in \mathsf{Stream}(\mathbb{S} \otimes \mathbb{X}, \mathbb{S} \otimes \mathbb{Y})$*, and* $g \in \mathsf{Stream}(\mathbb{X}', \mathbb{Y}')$*, we claim that*

$$\alpha \cdot \mathbf{fbk}^{S}(f \otimes g) = \mathbf{fbk}^{S}(f) \otimes g.$$

Proof. Consequence of Lemma 7.22, after applying the necessary coherence morphisms. □

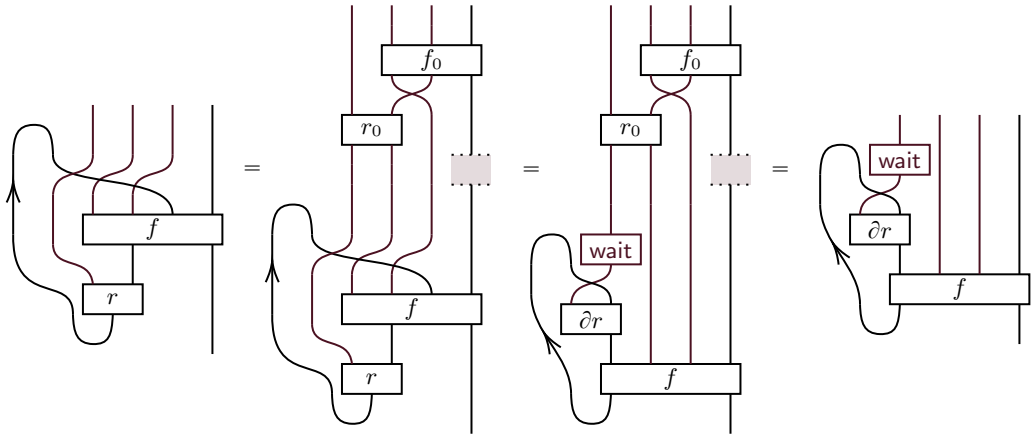


FIGURE 21. The sliding axiom (A5).

Lemma 7.24. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *with memories satisfies the sliding axiom (A5). Given two streams* $f \in \mathsf{Stream}(A \cdot \partial\mathbb{S} \otimes \mathbb{X}, \mathbb{T} \otimes \mathbb{Y})$*, and* $r \in \mathsf{Stream}(C \cdot \mathbb{T}, \mathbb{S})$ *we claim that, for each* $k \colon B \otimes Q \to C \otimes P$*,*

$$k \cdot \mathbf{fbk}^{\mathbb{S}}(f^{A\otimes P}; (r^C \otimes \mathrm{id})) = \mathbf{fbk}^{\mathbb{T}}(k \cdot (\partial r^C \otimes \mathrm{id}); f^{A\otimes P}).$$

Proof. First, we note that both sides of the equation (which, from now on, we call LHS and RHS, respectively) represent streams with different memories.

- $M(LHS) = M(f) \otimes M(r) \otimes S_0$,
- $M(RHS) = M(r) \otimes M(f) \otimes T_0$.

We will prove that they are related by dinaturality over the symmetry and the first action of r, that is, $\sigma \fatsemi (\mathsf{now}(r) \otimes \mathrm{id})$. We know that $\mathsf{now}(LHS) = \mathsf{now}(RHS) \fatsemi (\sigma \fatsemi (\mathrm{id} \otimes \mathsf{now}(r)))$ by string diagrams (see Figure 21). Using coinduction,

$$\begin{aligned}
&(\sigma \fatsemi \underline{\mathsf{now}(r)}) \cdot \mathsf{later}(LHS)\\
=&(\sigma \fatsemi \underline{\mathsf{now}(r)}) \cdot \mathbf{fbk}^{\mathbb{S}}(\mathsf{later}(f)^{M(f)\otimes S_0} \fatsemi (\mathsf{later}(r)^{M(r)} \otimes \mathrm{id}))\\
=&\mathbf{fbk}^{\mathbb{T}}((\sigma \fatsemi \underline{\mathsf{now}(r)}) \cdot (\partial\, \mathsf{later}(r)^{M(r)} \otimes \mathrm{id}) \fatsemi \mathsf{later}(f)^{M(f)\otimes S_0})\\
=&\mathbf{fbk}^{\mathbb{T}}(\sigma \cdot (\mathsf{later}(\partial r)^{M(r)} \otimes \mathrm{id}) \fatsemi \mathsf{later}(f)^{M(f)\otimes S_0})\\
=&\,\mathsf{later}(RHS),
\end{aligned}$$

we show that $(\sigma \fatsemi \underline{\mathsf{now}(r)}) \cdot \mathsf{later}(LHS) = \mathsf{later}(RHS)$. □

Lemma 7.25. *The structure* $(\mathsf{Stream}, \mathbf{fbk})$ *satisfies the sliding axiom (A5). Given two streams* $f \in \mathsf{Stream}(\partial\mathbb{S} \otimes \mathbb{X}, \mathbb{T} \otimes \mathbb{Y})$*, and* $r \in \mathsf{Stream}(\mathbb{T}, \mathbb{S})$ *we claim that,* $\mathbf{fbk}^{\mathbb{S}}(f \fatsemi (r \otimes \mathrm{id})) = \mathbf{fbk}^{\mathbb{T}}((\partial r \otimes \mathrm{id}) \fatsemi f)$.

Proof. Consequence of Lemma 7.24, after applying the necessary coherence morphisms. □

Theorem 7.26 (From Theorem 7.26)**.** *Monoidal streams over a productive symmetric monoidal category* $(\mathsf{C}, \otimes, I)$ *form a feedback monoidal category* $(\mathsf{Stream}, \mathbf{fbk})$ *over the functor* $\partial \colon [\mathbb{N}, \mathsf{C}] \to [\mathbb{N}, \mathsf{C}]$.

Proof. We have proven that it satisfies all the feedback axioms:

- the tightening axiom by Lemma 7.17,

- the vanishing axiom by Lemma 7.19,
- the joining axiom by Lemma 7.21,
- the strength axiom by Lemma 7.23,
- and the sliding axiom by Lemma 7.25.

Thus, it is a feedback structure. □

Corollary 7.27. *There is a "semantics" identity-on-objects feedback monoidal functor* $\mathbf{Sm} \colon \mathsf{St}_\partial[\mathbb{N}, \mathsf{C}] \to \mathsf{Stream}$ *from the free feedback monoidal category on the functor* $\hat{\partial} \colon [\mathbb{N}, \mathsf{C}] \to [\mathbb{N}, \mathsf{C}]$ *to the category of monoidal streams. Every dinatural stateful sequence* $\langle f_n \colon M_{n-1} \otimes X_n \to Y_n \otimes M_n \rangle_{n \in \mathbb{N}}$ *gives a monoidal stream* $\mathbf{Sm}(f)$*, which is defined by* $M(\mathbf{Sm}(f)) = M_0$*,*

$$\mathsf{now}(\mathbf{Sm}(f)) = f_0, \text{ and } \mathsf{later}(\mathbf{Sm}(f)) = \mathbf{Sm}(f^+),$$

and this is well-defined. Moreover, this functor is full when C *is productive; it is not generally faithful.*

Proof. We construct $\mathbf{Sm}$ from Theorems 5.6 and 7.26. Moreover, when C is productive, by Theorem 6.19, monoidal streams are dinatural sequences quotiented by observational equivalence, giving the fullness of the functor. □

8. Cartesian Streams

Dataflow languages such as Lucid or Lustre [WA$^+$85, HLR92] can be thought of as using an underlying cartesian monoidal structure: we can copy and discard variables and resources without affecting the order of operations. These abilities correspond exactly to cartesianness thanks to Fox's theorem (Theorem 8.3, see [Fox76]).

8.1. Causal stream functions.

In the cartesian case, there is available literature on the categorical semantics of dataflow programming languages [BCLH93, UV08, Cou19, Del19, Oli84]. Uustalu and Vene [UV05] provide elegant comonadic semantics to a Lucid-like programming language using the non-empty list comonad. In their framework, *streams* with types $\mathbb{X} = (X_0, X_1, \dots)$ are families of elements $\mathbf{1} \to X_n$. *Causal stream functions* from $\mathbb{X} = (X_0, X_1, \dots)$ to $\mathbb{Y} = (Y_0, Y_1, \dots)$ are families of functions $f_n \colon X_0 \times \cdots \times X_n \to Y_n$. Equivalently, they are, respectively, the states $(\mathbb{1} \to \mathbb{X})$ and morphisms $(\mathbb{X} \to \mathbb{Y})$ of the cokleisli category of the comonad $\mathbf{List}^+ \colon [\mathbb{N}, \mathsf{Set}] \to [\mathbb{N}, \mathsf{Set}]$ defined by

$$(\mathbf{List}^+(\mathbb{X}))_n \coloneqq \coprod_{i=0}^{n} X_i.$$

The functor underlying this comonad can be defined on all symmetric monoidal categories.

Definition 8.1. Let $(\mathsf{C}, \otimes, I)$ be a symetric monoidal category. There is a functor $\mathbf{List}^+ \colon \mathsf{C} \to \mathsf{C}$ defined on objects by

$$\mathbf{List}^+(X)_n \coloneqq \bigotimes_{i=0}^{n} X_i.$$

This functor is oplax monoidal with the natural transformations $\psi_0^+ \colon \mathbf{List}^+(I) \to I$ and $\psi_{X,Y} \colon \mathbf{List}^+(X \otimes Y) \to \mathbf{List}^+(X) \otimes \mathbf{List}^+(Y)$ given by symmetries, associators and unitors.

The comonad structure of $\mathbf{List}^+$, however, can be extended to other base categories *only* as long as C is cartesian. More precisely, the structure of $\mathbf{List}^+$ on cartesian categories is that of an *oplax monoidal comonad*.

Definition 8.2 (Oplax monoidal comonad)**.** In a monoidal category $(\mathsf{C}, \otimes, I)$, a comonad (R, ε, δ) is an *oplax monoidal comonad* when the endofunctor $R\colon \mathsf{C} \to \mathsf{C}$ is oplax monoidal with laxators $\psi_{X,Y}\colon R(X \otimes Y) \to RX \otimes RY$ and $\psi^I\colon RI \to I$, and both the counit $\varepsilon_X\colon RX \to X$ and the comultiplication $\delta_X\colon RX \to RRX$ are monoidal natural transformations.

Explicitly, the comonad structure needs to satisfy: $\varepsilon_I = \psi_0$, $\varepsilon_{X\otimes Y} = \psi_{X,Y} \fatsemi (\varepsilon_X \otimes \varepsilon_Y)$, $\delta_I \fatsemi \psi_0 \fatsemi \psi_0 = \psi_0$ and $\delta_{X\otimes Y} \fatsemi \psi_{X,Y} \fatsemi \psi_{RX,RY} = \psi_{X,Y} \fatsemi (\delta_X \otimes \delta_Y)$.

Alternatively, an *oplax monoidal comonad* is a comonoid in the bicategory **MonOplax** of oplax monoidal functors with composition and monoidal natural transformations.

Indeed, we can prove (Theorem 8.6) that the mere existence of such a comonad implies cartesianness of the base category. For this, we introduce a refined version of Fox's theorem (Theorem 8.5).

8.2. **Fox's theorem.** Fox's theorem [Fox76] is a classical characterisation of cartesian monoidal categories in terms of the existence of a uniform cocommutative comonoid structure on all objects of a monoidal category.

Theorem 8.3 (Fox's theorem [Fox76])**.** *A symmetric monoidal category $(\mathsf{C}, \otimes, I)$ is cartesian monoidal if and only if every object $X \in \mathsf{C}$ has a cocommutative comonoid structure $(X, \varepsilon_X, \delta_X)$, every morphism of the category $f\colon X \to Y$ is a comonoid homomorphism, and this structure is uniform across the monoidal category, meaning that $\varepsilon_{X\otimes Y} = \varepsilon_X \otimes \varepsilon_Y$, that $\varepsilon_I = \mathrm{id}$, that $\delta_I = \mathrm{id}$ and that $\delta_{X\otimes Y} = (\delta_X \otimes \delta_Y) \fatsemi (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id})$.*

Most sources ask the comonoid structure in Fox's theorem (Theorem 8.3) to be cocommutative [Fox76, FS19]. However, cocommutativity and coassociativity of the comonoid structure are implied by uniformity and naturality of the structure. We present an original refined version of Fox's theorem, which asks for a counital comagma structure on every object.

Definition 8.4. A *comagma* in a monoidal category C is a pair (X, δ) of an object X of C and a morphism $\delta\colon X \to X \otimes X$. A comagma is *counital* if, additionally, we specify a morphism $\varepsilon\colon X \to I$ and this makes δ unital: $\delta \fatsemi (\mathsf{id} \otimes \varepsilon) = \mathsf{id} = \delta \fatsemi (\varepsilon \otimes \mathsf{id})$ (Figure 22, left). A *homomorphism* of counital comagmas $f\colon (X, \varepsilon_X, \delta_X) \to (Y, \varepsilon_Y, \delta_Y)$ is a morphism $f\colon X \to Y$ that preserves the counit and the comultiplication: $f \fatsemi \delta_Y = \delta_X \fatsemi (f \otimes f)$ and $f \fatsemi \varepsilon_Y = \varepsilon_X$ (Figure 22, right).

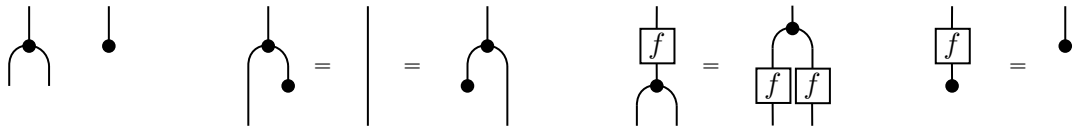


FIGURE 22. Structure and axioms of a counital comagma (left) and axioms of counital comagma homomorphism (right).

Theorem 8.5 (Refined Fox's theorem, c.f. [Mel09])**.** *A symmetric monoidal category* $(\mathsf{C}, \otimes, I)$ *is cartesian monoidal if and only if every object* $X \in \mathsf{C}$ *has a counital comagma structure* $(X, \varepsilon_X, \delta_X)$*, or* $(X, \text{!}_X, \text{⅄}_X)$*, every morphism of the category* $f\colon X \to Y$ *is a comagma homomorphism, and this structure is uniform across the monoidal category: meaning that* $\varepsilon_{X\otimes Y} = \varepsilon_X \otimes \varepsilon_Y$*,* $\varepsilon_I = \mathrm{id}$*,* $\delta_I = \mathrm{id}$ *and* $\delta_{X\otimes Y} = (\delta_X \otimes \delta_Y); (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id})$*.*

Proof. We prove that such a comagma structure is necessarily coassociative and cocommutative. Note that any comagma homomorphism $f\colon A \to B$ must satisfy $\delta_A \fatsemi (f \otimes f) = f \fatsemi \delta_B$. In particular, $\delta_X\colon X \to X \otimes X$ must itself be a comagma homomorphism (see Figure 23), meaning that

$$\delta_X \fatsemi (\delta_X \otimes \delta_X) = \delta_X \fatsemi \delta_{X\otimes X} = \delta_X \fatsemi (\delta_X \otimes \delta_X) \fatsemi (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id}), \tag{8.1}$$

where the second equality follows by uniformity.

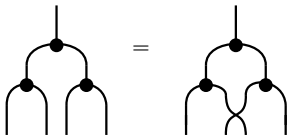


FIGURE 23. Comultiplication is a comagma homomorphism.

Now, we prove cocommutativity (Figure 24): composing both sides of Equation (8.1) with $(\epsilon_X \otimes \mathrm{id} \otimes \mathrm{id} \otimes \epsilon_X)$ discards the two external outputs and gives $\delta_X = \delta_X \fatsemi \sigma_X$.

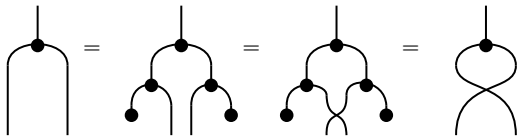


FIGURE 24. Cocommutativity

Now, we prove coassociativity (Figure 25): composing both sides of Equation (8.1) with $(\mathrm{id} \otimes \epsilon_X \otimes \mathrm{id} \otimes \mathrm{id})$ discards one of the middle outputs and gives $\delta_X \fatsemi (\mathrm{id} \otimes \delta_X) = \delta_X \fatsemi (\delta_X \otimes \mathrm{id})$.

A coassociative and cocommutative comagma is a cocommutative comonoid. We can then apply the classical form of Fox's theorem (Theorem 8.3). □

One could hope to add effects such as probability or non-determinism to set-based streams via the bikleisli category arising from a monad-comonad distributive law $\mathbf{List}^+ \circ \mathbb{T} \Rightarrow \mathbb{T} \circ \mathbf{List}^+$ [PW99, Bec69], as proposed by Uustalu and Vene [UV05]. This would correspond to a lifting of the $\mathbf{List}^+$ comonad to the kleisli category of some commutative monad T; the arrows $\mathbb{X} \to \mathbb{Y}$ of such a category would look as follows,

$$f_n\colon X_1 \times \cdots \times X_n \to TY_n.$$

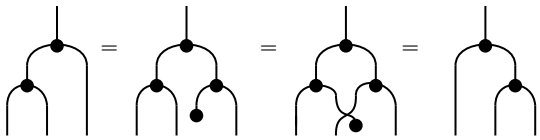


FIGURE 25. Coassociativity

However, we show that this would not result in a monoidal comonad whenever $\mathsf{kl}(T)$ is not cartesian. To see explicitly what fails, we use the string diagrams to show how composition would not work for the case $n = 2$ (Figure 26). This composition is not associative or unital whenever the kleisli category does not have natural comultiplications or counits, respectively (Figure 27).

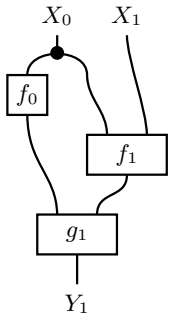


FIGURE 26. Composition in the case $n = 2$.

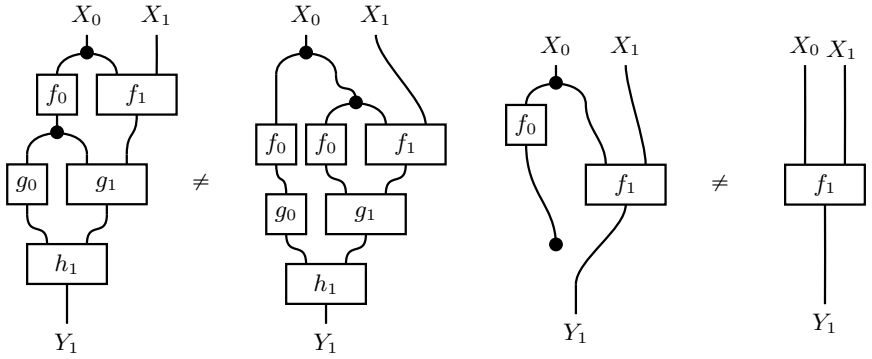


FIGURE 27. *Left*: Failure of associativity if copying is not natural, as it happens, for instance, with stochastic functions. *Right*: Failure of unitality if discarding is not natural, as it happens, for instance, with partial functions.

Theorem 8.6. *The oplax monoidal functor* $\mathbf{List}^+$ *has an oplax monoidal comonad structure if and only if its base monoidal category* $(\mathsf{C}, \otimes, I)$ *is cartesian monoidal.*

Proof. When C is cartesian, we can construct the comonad structure using projections $\prod_{i=0}^{n} X_i \to X_n$ and copying together with symmetries $\prod_{i=0}^{n} X_i \to \prod_{i=0}^{n} \prod_{k=0}^{i} X_k$. These are monoidal natural transformations making $\mathbf{List}^+$ an oplax monoidal comonad.

Suppose (L, ε, δ) is an oplax monoidal comonad structure. This means it has families of natural transformations

$$\delta_n \colon \bigotimes_{i=0}^{n} X_i \to \bigotimes_{i=0}^{n} \bigotimes_{k=0}^{i} X_k \text{ and } \varepsilon_n \colon \bigotimes_{i=0}^{n} X_i \to X_n.$$

We use these to construct a uniform counital comagma structure on every object of the category. By the refined version of Fox's theorem (Theorem 8.5), this implies that C is cartesian monoidal.

Let $X \in \mathsf{C}$ be any object. Choosing $n = 2$, $X_0 = X$ and $X_1 = I$; and using coherence maps, we get $\delta_2 \colon X \to X \otimes X$ and $\varepsilon_2 \colon X \to I$. These are coassociative, counital, natural and uniform because the corresponding transformations δ and ε are themselves coassociative, counital, natural and monoidal. This induces a uniform comagma structure in every object $(X, \delta_2, \varepsilon_2)$; with this structure, every morphism of the category is a comagma homomorphism because δ_2 and ε_2 are natural. □

This result implies that we cannot directly extend Uustalu and Vene's approach to non-cartesian monoidal structures. However, we prove in the next section that our definition of monoidal streams particularizes to their *causal stream functions* [UV05, SK19].

8.3. **Cartesian monoidal streams.** The main claim of this section is that, in a cartesian monoidal category, monoidal streams instantiate to *causal stream functions* (Theorem 8.9). Let us fix such a category, $(\mathsf{C}, \times, 1)$.

The first observation is that the universal property of the cartesian product simplifies the fixpoint equation that defines monoidal streams. This is a consequence of the following chain of isomorphisms, where we apply a Yoneda reduction to simplify the coend.

$$
\begin{aligned}
& \mathsf{Stream}(\mathbb{X}, \mathbb{Y}) \\
\cong \quad & \text{(By definition)} \\
& \int^{M} \hom(X_0, M \times Y_0) \times \mathsf{Stream}(M \cdot \mathbb{X}^+, \mathbb{Y}) \\
\cong \quad & \text{(Universal property of the product)} \\
& \int^{M} \hom(X_0, M) \times \hom(X_0, Y_0) \times \mathsf{Stream}(M \cdot \mathbb{X}^+, \mathbb{Y}) \\
\cong \quad & \text{(Yoneda reduction)} \\
& \hom(X_0, Y_0) \times \mathsf{Stream}(X_0 \cdot \mathbb{X}^+, \mathbb{Y}^+).
\end{aligned}
$$

Explicitly, the Yoneda reduction works as follows: the first action of a stream $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ can be uniquely split as $\mathsf{now}(f) = (f_1, f_2)$ for some $f_1 \colon X_0 \to Y_0$ and $f_2 \colon X_0 \to M(f)$. Under the *dinaturality* equivalence relation, $(\sim)$, we can always find a unique representative with $M = X_0$ and $f_2 = \mathrm{id}_{X_0}$.

Proposition 8.7. *Cartesian monoidal categories are productive.*

Proof. Let $\langle \alpha | \in \mathbf{Stage}_1(\mathbb{X}, \mathbb{Y})$. For some given representative $\alpha \colon X_0 \to M \otimes Y_0$, we define the two projections $\alpha_Y = \alpha \fatsemi \underline{\blacklozenge}_M \colon X_0 \to Y_0$ and $\alpha_M = \alpha \fatsemi \underline{\blacklozenge}_Y$. The second projection α_M depends on the specific representative α we have chosen; however, the first projection α_Y is defined independently of the specific representative α, as a consequence of naturality of the discarding map (see Fox's theorem for cartesian monoidal categories Theorem 8.3). We define $\alpha_0 = \delta_{X_0} \fatsemi \underline{\alpha_Y}$. Then, we can factor any representative as $\alpha = \alpha_0; \underline{\alpha_M}$ (see Figure 28). Now, assume that we have two representatives $\langle \alpha_i | = \langle \alpha_j |$ for which $\langle \alpha_i \fatsemi u | = \langle \alpha_j \fatsemi v |$. By naturality of the discarding map, $\alpha_i \fatsemi \blacklozenge = \alpha_j \fatsemi \blacklozenge$, and we call this map α_Y. Again by naturality of the discarding map, $\alpha_i \fatsemi u \fatsemi \blacklozenge = \alpha_j \fatsemi v \fatsemi \blacklozenge$, and discarding the output in Y, we get that $\alpha_{M,i} \fatsemi u \fatsemi \blacklozenge = \alpha_{M,j} \fatsemi v \fatsemi \blacklozenge$, which implies $\langle \alpha_{M,i} \fatsemi u | = \langle \alpha_{M,j} \fatsemi v |$. □

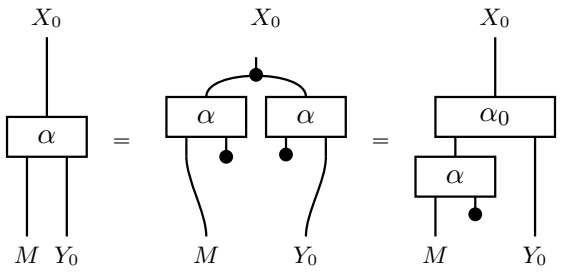


FIGURE 28. Productivity for cartesian categories.

The definition of monoidal streams in the cartesian case is thus simplified (Definition 8.8). From there, the explicit construction of cartesian monoidal streams is straightforward.

Definition 8.8 (Cartesian monoidal streams)**.** The set of *cartesian monoidal streams*, given inputs $\mathbb{X}$ and outputs $\mathbb{Y}$, is the terminal fixpoint of the equation

$$\mathsf{Stream}(\mathbb{X}, \mathbb{Y}) \cong \hom(X_0, Y_0) \times \mathsf{Stream}(X_0 \cdot \mathbb{X}^+, \mathbb{Y}^+).$$

In other words, a cartesian monoidal stream $f \in \mathsf{Stream}(\mathbb{X}, \mathbb{Y})$ is a pair consisting of

- $\mathsf{fst}(f) \in \hom(X_0, Y_0)$, the *first action*, and
- $\mathsf{snd}(f) \in \mathsf{Stream}(X_0 \cdot \mathbb{X}^+, \mathbb{Y}^+)$, the *rest of the action*.

Theorem 8.9. *In the cartesian case, the final fixpoint of the equation in Figure 11 is given by the set of causal functions,*

$$\mathsf{Stream}(\mathbb{X}, \mathbb{Y}) = \prod_{n \in \mathbb{N}}^{\infty} \hom(X_0 \times \cdots \times X_n, Y_n).$$

That is, the category Stream *of monoidal streams coincides with the cokleisli monoidal category of the non-empty list monoidal comonad* $\mathbf{List}^+ \colon [\mathbb{N}, \mathsf{C}] \to [\mathbb{N}, \mathsf{C}]$.

Proof. By Adamek's theorem (Theorem 2.5). □

Corollary 8.10. *Let* $(\mathsf{C}, \times, \mathbf{1})$ *be a cartesian monoidal category. The category* Stream *is cartesian monoidal.*

Remark 8.11. The feedback operation in cartesian streams is similar to the delayed trace of stateful morphism sequences described by Sprunger and Katsumata [SK19]. In fact, extensional equivalence classes of stateful morphism sequences over a cartesian category C are in bijection with cartesian streams over the same category C. Explicitly, for a stateful morphism sequence, $(i, \mathbb{s}) \colon \mathbb{X} \to \mathbb{Y}$, the corresponding cartesian stream is $i \cdot \mathbb{s}$, of the same type. Vice versa, a cartesian stream $\mathbb{f} \colon \mathbb{X} \to \mathbb{Y}$ gives an equivalence class of stateful morphism sequences $(\mathsf{id}_1, \mathbb{f})$ of the same type. These mappings give isomorphisms when stateful morphism sequences are quotiented by extensional equivalence as defined by Sprunger and Katsumata [SK19].

Moreover, delayed trace and feedback are interderivable. Following Sprunger and Katsumata [SK19], consider the operation $\bigcirc$ that removes the first component of the stream — that yields $\bigcirc\mathbb{T} = (T_1, T_2, \dots)$ for a stream $\mathbb{T} = (T_0, T_1, \dots)$. The delay ∂ is its left inverse, $\bigcirc\partial\mathbb{T} = \mathbb{T}$. For a stateful morphism sequence $(i, \mathbb{s}) \colon \mathbb{T} \times \mathbb{X} \to \bigcirc\mathbb{T} \times \mathbb{Y}$, its delayed trace along $\mathbb{T}$ with initial point p can be expressed as a feedback along $\mathbb{T}^+$. Vice versa, the feedback of $\mathbb{f} \colon \partial\mathbb{U} \times \mathbb{X} \to \mathbb{U} \times \mathbb{Y}$ along $\mathbb{U}$ can be expressed as a delayed trace along $\partial\mathbb{U}$.

$$\mathsf{tr}_p^{\mathbb{T}}(i, \mathbb{s}) = \mathbf{fbk}^{\mathbb{T}^+}((i \times p) \cdot \mathbb{s}), \qquad \mathbf{fbk}^{\mathbb{U}}(\mathbb{f}) = \mathsf{tr}_{\mathsf{id}_1}^{\partial\mathbb{U}}(\mathbb{f}).$$

We restrict our discussion to the cartesian case studied by Sprunger and Katsumata [SK19], but a similar correspondence would hold generalising stateful morphism sequences to the monoidal case (cf. [CdVP21]); in general, the equivalence relation of monoidal streams does not coincide with the one of monoidal stateful morphism sequences.

8.4. **Example: the Fibonacci sequence.** Consider $(\mathsf{Set}, \times, \mathbf{1})$, the cartesian monoidal category of small sets and functions. And let us go back to the morphism $\mathsf{fib} \in \mathsf{Stream}(\mathbf{1}, \mathbb{N})$ that we presented in the Introduction (Figure 2). By Theorem 8.9, a morphism of this type is, equivalently, a sequence of natural numbers. Using the previous definitions in Sections 7 and 8, we can explicitly compute this sequence to be $\mathsf{fib} = [0, 1, 1, 2, 3, 5, 8, \dots]$ (see the implementation [Rom22]).

9. Stochastic Streams

Monoidal categories are well suited for reasoning about probabilistic processes. Several different categories of probabilistic channels have been proposed in the literature [Pan99, BFP16, CJ19]. They were largely unified by Fritz [Fri20] under the name of *Markov categories*. For simplicity, we work in the discrete stochastic setting, i.e. in the Kleisli category of the finite distribution monad, Stoch, but we will be careful to isolate the relevant structure of Markov categories that we use.

The main result of this section is that *controlled stochastic processes* [FR75, Ros96] are precisely monoidal streams over Stoch. That is, controlled stochastic processes are the canonical solution over Stoch of the fixpoint equation in Figure 11.

9.1. Preliminaries: Markov categories.

Definition 9.1 (Markov category, [Fri20, Definition 2.1])**.** A *Markov category* C is a symmetric monoidal category in which each object $X \in \mathsf{C}$ has a cocommutative comonoid structure $(X, \varepsilon = \text{!}_X : X \to I, \delta = \text{¡}_X : X \to X \otimes X)$ with

- ▷ *uniform* comultiplications, $\text{¡}_{X\otimes Y} = (\text{¡}_X \otimes \text{¡}_Y) \fatsemi \sigma_{X,Y}$;
- ▷ *uniform* counits, $\text{!}_{X\otimes Y} = \text{!}_X \otimes \text{!}_Y$; and
- ▷ *natural* counits, $f \fatsemi \text{!}_Y = \text{!}_X$ for each $f : X \to Y$.

Crucially, comultiplications do not need to be natural.

Definition 9.2. The finite distribution commutative monad $\mathbf{D} : \mathsf{Set} \to \mathsf{Set}$ associates to each set the set of finite-support probability distributions over it.

$$\mathbf{D}(X) = \left\{ p : X \to [0,1] \;\middle|\; \sum_{p(x)>0}^{|\{x|p(x)>0\}|<\infty} p(x) = 1 \right\}.$$

We call Stoch the symmetric monoidal Kleisli category of the finite distribution monad, $\mathsf{kl}(\mathbf{D})$. We write $f(y|x)$ for the probability $f(x)(y) \in [0,1]$. Composition, $(f \fatsemi g)$, is defined by

$$(f \fatsemi g)(z|x) = \sum_{y \in Y} g(z|y) f(y|x).$$

The cartesian product ($\times$) in Set induces a monoidal (non-cartesian) product in Stoch. That is, Stoch has comonoids $(\text{🕷})_X : X \to X \times X$ on every object, with $(\text{🌡})_X \colon X \to 1$ as counit. However, contrary to what happens in Set, these comultiplications are not natural: *sampling and copying the result* is different from *taking two independent samples*. The Markov category Stoch also has *conditionals* [Fri20], a property which we will use to prove the main result regarding stochastic processes.

Remark 9.3 [Fri20, Remark 2.4]**.** Any cartesian category is a Markov category. However, not any Markov category is cartesian, and the most interesting examples are those that fail to be cartesian, such as Stoch. The failure of comultiplication being natural makes it impossible to apply Fox's theorem (Theorem 8.3).

Remark 9.4 (Triangle notation)**.** In a Markov category, given any $f \colon X_0 \to Y_0$ and any $g \colon Y_0 \otimes X_0 \otimes X_1 \to Y_1$, we write $(f \lhd g) \colon X_0 \otimes X_1 \to Y_0 \otimes Y_1$ for the morphism defined by

$$(f \lhd g) = \underline{(\text{🕷}_{X_0}) \fatsemi f \fatsemi (\text{🕷}_{Y_0})} \fatsemi g,$$

which is the string diagram in Figure 29.

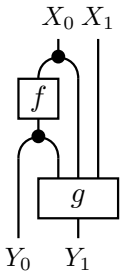


FIGURE 29. The morphism $(f \lhd g)$.

Proposition 9.5 (Triangle notation is associative)**.** *Up to symmetries,*

$$(f \lhd g) \lhd h = f \lhd (g \lhd h).$$

We may simply write $(f \lhd g \lhd h)$ for any of the two, omitting the symmetry.

Proof. Using string diagrams (Figure 30). Note that (🕷) is coassociative and cocommutative. □

The structure of a Markov category is very basic. In most cases, we do need extra structure to reason about probabilities: this is the role of conditionals and ranges.

Definition 9.6 (Conditionals, [Fri20, Definition 11.5])**.** Let C be a Markov category. We say that C has *conditionals* if for every morphism $f \colon A \to X \otimes Y$, writing $f_Y \colon A \to X$ for its first projection, there exists $c_f \colon X \otimes A \to Y$ such that $f = f_Y \lhd c_f$ (Figure 31).

Proposition 9.7. *The Markov category* Stoch *has conditionals* [Fri20, Example 11.6].

Proof. Let $f \colon A \to X \otimes Y$. If Y is empty, we are automatically done. If not, pick some $y_0 \in Y$, and define

$$c_f(y|x,a) = \begin{cases} f(x,y|a) / \sum_{x \in X} f(x,y|a) \\ \qquad \text{if } f(x,y|a) > 0 \text{ for some } x \in X, \\ (y = y_0) \quad \text{otherwise.} \end{cases}$$

It is straightforward to check that this does indeed define a distribution, and that it factors the original f as expected. □

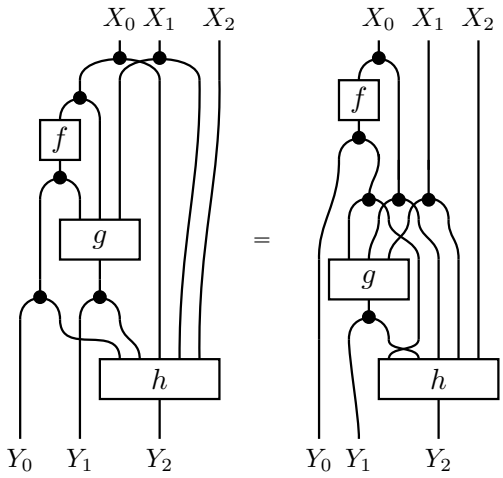


FIGURE 30. Associativity, up to symmetries, of the triangle operation.

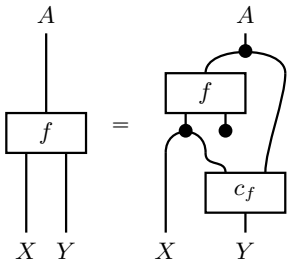


FIGURE 31. Condititionals in a Markov category.

Definition 9.8 (Ranges)**.** In a Markov category, a *range* for a morphism $f\colon A \to B$ is a morphism $r_f\colon A \otimes B \to A \otimes B$ that

(1) does not change its output $f \vartriangleleft \mathrm{id}_{A\otimes B} = f \vartriangleleft r_f$,
(2) is *deterministic*, meaning $r_f \fatsemi \curlywedge_{A\otimes B} = \curlywedge_{A\otimes B} \fatsemi (r_f \otimes r_f)$,
(3) and has the *range* property, $f \vartriangleleft g = f \vartriangleleft h$ must imply

$$(r_f \otimes \mathrm{id}) \fatsemi g = (r_f \otimes \mathrm{id}) \fatsemi h$$

for any suitably typed g and h.

We say that a Markov category *has ranges* if there exists a range for each morphism of the category.

Proposition 9.9. *The Markov category* Stoch *has ranges.*

Proof. Given $f\colon A \to B$, we know that for each $a \in A$ there exists some $b_a \in B$ such that $f(b_a|a) > 0$. We fix such $b_a \in B$, and we define $r_f\colon A \otimes B \to A \otimes B$ as

$$r_f(a,b) = \begin{cases} (a,b) & \text{if } f(b|a) > 0, \\ (a,b_a) & \text{if } f(b|a) = 0. \end{cases}$$

It is straightforward to check that it satisfies all the properties of ranges. □

Remark 9.10. There already exists a notion of categorical *range* in the literature, due to Cockett, Guo and Hofstra [CGH12]. It arises in parallel to the notion of *support* in *restriction categories* [CL02]. The definition better suited for our purposes is different, even

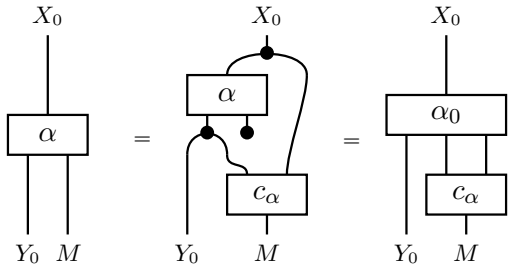


FIGURE 32. Productivity for Markov categories.

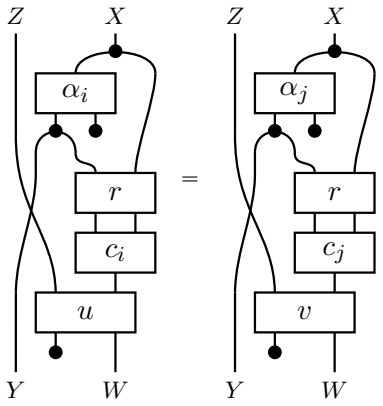


FIGURE 33. Applying the properties of range.

if it seems inspired by the same idea. The main difference is that we are using a *controlled range*; that is, the range of a morphism depends on the input to the original morphism. We keep the name hoping that it will not cause any confusion, as we do not deal explicitly with restriction categories in this text.

Theorem 9.11. *Any Markov category with conditionals and ranges is productive.*

Proof. Given any $\langle \alpha | \in \mathbf{Stage}_1(\mathbb{X}, \mathbb{Y})$, we can define

$$\alpha_0 = \curlywedge_{X_0} \fatsemi \underline{\alpha \fatsemi (\curlywedge_{Y_0} \otimes \downarrow_M)}.$$

This is indeed well-defined because of naturality of the discarding map $(\downarrow)_M \colon M \to I$ in any Markov category. Let $c_\alpha \colon Y_0 \otimes X_0 \to M$ be a conditional of α. This representative can then be factored as $\alpha = \alpha_0 \fatsemi \underline{c_\alpha}$ (Figure 32).

Now assume that for two representatives $\langle \alpha_i | = \langle \alpha_j |$ we have that $\langle \alpha_i \fatsemi u | = \langle \alpha_j \fatsemi v |$. By naturality of the discarding, $\alpha_i \fatsemi \underline{\varepsilon} = \alpha_j \fatsemi \underline{\varepsilon}$, and let r be a range of this map. Again by naturality of discarding, we have $\underline{\alpha_i} \fatsemi \underline{u} \fatsemi \underline{\downarrow} = \underline{\alpha_j} \fatsemi \underline{v} \fatsemi \underline{\downarrow}$. Let then c_i and c_j be conditionals of α_i and α_j: we have that $(\alpha_0 \lhd c_i) \fatsemi u \fatsemi \downarrow = (\alpha_0 \lhd c_j) \fatsemi v \fatsemi \downarrow$. By the properties of ranges (Figure 33), $(\alpha_0 \lhd r \fatsemi c_i) \fatsemi u \fatsemi \downarrow_{M(u)} = (\alpha_0 \lhd r \fatsemi c_j) \fatsemi v \fatsemi \downarrow_{M(v)}$, and thus, $r \fatsemi c_i \fatsemi u \fatsemi \downarrow_{M(u)} = r \fatsemi c_j \fatsemi v \fatsemi \downarrow_{M(v)}$. We pick $s_i = r \fatsemi c_i$ and we have proven that $\langle \underline{s_i} \fatsemi u | = \langle \underline{s_j} \fatsemi v |$. □

9.2. **Stochastic processes.** We start by recalling the notion of *stochastic process* from probability theory and its "controlled" version. The latter is used in the context of *stochastic control* [FR75, Ros96], where the user has access to the parameters or optimization variables of a probabilistic model.

A *discrete stochastic process* is defined as a collection of random variables $Y_1, \dots Y_n$ indexed by discrete time. At any time step n, these random variables are distributed according to some $p_n \in \mathbf{D}(Y_1 \times \cdots \times Y_n)$. Since the future cannot influence the past, the marginal of p_{n+1} over Y_{n+1} must equal p_n. When this occurs, we say that the family of distributions $(p_n)_{n\in\mathbb{N}}$ is *causal*.

More generally, there may be some additional variables $X_1, \dots, X_n$ which we have control over. In this case, a *controlled stochastic process* is defined as a collection of *controlled random variables* distributing according to $f_n \colon X_1 \times \cdots \times X_n \to \mathbf{D}(Y_1, \dots, Y_n)$. Causality ensures that the marginal of f_{n+1} over Y_{n+1} must equal f_n.

Definition 9.12. Let $\mathbb{X}$ and $\mathbb{Y}$ be sequences of sets. A *controlled stochastic process* $f \colon \mathbb{X} \to \mathbb{Y}$ is a sequence of functions

$$f_n \colon X_0 \times \cdots \times X_n \to \mathbf{D}(Y_0 \times \cdots \times Y_n)$$

satisfying *causality* (the *marginalisation property*). That is, such that f_n coincides with the marginal distribution of f_{n+1} on the first n variables, $f_{n+1} \fatsemi \mathbf{D}\pi_{Y_0,\dots,Y_n} = \pi_{X_0,\dots,X_n} \fatsemi f_n$, making the diagram in Figure 34 commute.

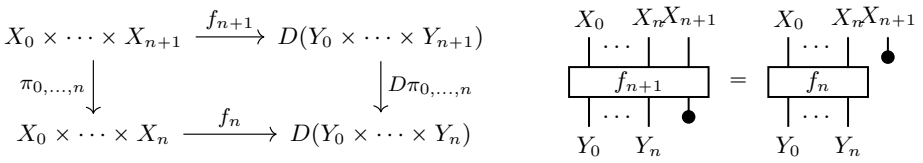


FIGURE 34. Marginalisation for stochastic processes.

Controlled stochastic processes with componentwise composition, identities and tensoring, are the morphisms of a symmetric monoidal category StochProc.

Proposition 9.13 (Factoring as conditionals)**.** *A stochastic process $f \colon \mathbb{X} \to \mathbb{Y}$ can be always written as*

$$f_n = c_0 \lhd c_1 \lhd \cdots \lhd c_n,$$

for some family of functions

$$c_n \colon Y_0 \times \cdots \times Y_{n-1} \times X_0 \times \cdots \times X_n \to Y_n,$$

called the conditionals *of the stochastic process.*

Proof. We proceed by induction, noting first that $c_0 = f_0$. In the general case, we apply conditionals to rewrite $f_{n+1} = (f_{n+1}; (!)_{Y_{n+1}}) \lhd c_{n+1}$. Because of the marginalization property, we know that $f_{n+1}; (!)_{Y_{n+1}} = f_n$. So finally, $f_{n+1} = f_n \lhd c_{n+1}$, which by the induction hypothesis gives the desired result. □

Proposition 9.14. *If two families of conditionals give rise to the same stochastic process,*

$$c_0 \lhd c_1 \lhd \cdots \lhd c_n = c'_0 \lhd c'_1 \lhd \cdots \lhd c'_n,$$

then, they also give rise to the same n-stage processes in Stoch*,*

$$\langle c_0 \lhd \mathrm{id} | c_1 \lhd \mathrm{id} | \dots | c_n \lhd \mathrm{id} | = \langle c'_0 \lhd \mathrm{id} | c'_1 \lhd \mathrm{id} | \dots | c'_n \lhd \mathrm{id} | .$$

Proof. We start by defining a family of morphisms r_n by induction. We take $r_0 = \mathrm{id}$ and r_{n+1} to be a *range* of $r_n; \copy; c_n$.

Let us prove now that for any $n \in \mathbb{N}$ and $i \leqslant n$,

$$r_i \fatsemi \blacktriangle \fatsemi c_i \lhd \cdots \lhd c_n = r_i \fatsemi \blacktriangle \fatsemi c'_i \lhd \cdots \lhd c'_n.$$

We proceed by induction. Observing that $c_0 = c'_0$, we prove it for $n = 0$ and also for the case $i = 0$ for any $n \in \mathbb{N}$. Assume we have it proven for n, so in particular we know that $r_i \fatsemi \blacktriangle \fatsemi c_i = r_i \fatsemi \blacktriangle \fatsemi c'_i$ for any $i \leqslant n$. Now, by induction on i, we can use the properties of ranges to show that

$$\begin{aligned}
r_i \fatsemi \blacktriangle \fatsemi c_i \lhd \cdots \lhd c_n &= r_i \fatsemi \blacktriangle \fatsemi c'_i \lhd \cdots \lhd c'_n \\
(r_i \fatsemi \blacktriangle \fatsemi c_i \lhd \mathrm{id}) \fatsemi c_{i+1} \lhd \cdots \lhd c_n &= (r_i \fatsemi \blacktriangle \fatsemi c'_i \lhd \mathrm{id}) \fatsemi c'_{i+1} \lhd \cdots \lhd c'_n \\
(r_i \fatsemi \blacktriangle \fatsemi c_i \lhd r_{i+1}) \fatsemi c_{i+1} \lhd \cdots \lhd c_n &= (r_i \fatsemi \blacktriangle \fatsemi c'_i \lhd r_{i+1}) \fatsemi c'_{i+1} \lhd \cdots \lhd c'_n \\
r_{i+1} \fatsemi \blacktriangle \fatsemi c_{i+1} \lhd \cdots \lhd c_n &= r_{i+1} \fatsemi \blacktriangle \fatsemi c'_{i+1} \lhd \cdots \lhd c'_n.
\end{aligned}$$

In particular, $r_n \fatsemi \blacktriangle \fatsemi c_n = r_n \fatsemi \blacktriangle \fatsemi c'_n$.

Now, we claim the following for each $n \in \mathbb{N}$ and each $i \leqslant n$,

$$\begin{aligned}
&\langle c_0 \lhd \mathrm{id}| \dots |c_n \lhd \mathrm{id}| = \\
&\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i(c_i \lhd \mathrm{id})|c_{i+1}| \dots |c_n \lhd \mathrm{id}|.
\end{aligned}$$

It is clear for $n = 0$ and for $i = 0$. In the inductive case for i,

$$\begin{aligned}
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i(c_i \lhd \mathrm{id})|c_{i+1} \lhd \mathrm{id}| \dots |c_n \lhd \mathrm{id}| &= \\
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i \fatsemi \blacktriangle \fatsemi c_i \lhd \mathrm{id}|c_{i+1} \lhd \mathrm{id}| \dots |c_n \lhd \mathrm{id}| &= \\
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i \fatsemi \blacktriangle \fatsemi c_i \lhd r_{i+1}|c_{i+1} \lhd \mathrm{id}| \dots |c_n \lhd \mathrm{id}| &= \\
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i \fatsemi \blacktriangle \fatsemi c_i \lhd \mathrm{id}|r_{i+1}(c_{i+1} \lhd \mathrm{id})| \dots |c_n \lhd \mathrm{id}| &= \\
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_i(c_i \lhd \mathrm{id})|r_{i+1}(c_{i+1} \lhd \mathrm{id})| \dots |c_n \lhd \mathrm{id}|.
\end{aligned}$$

A particular case of this claim is then that

$$\begin{aligned}
\langle c_0 \lhd \mathrm{id}| \dots |c_n \lhd \mathrm{id}| &= \\
\langle r_0(c_0 \lhd \mathrm{id})| \dots |r_n(c_n \lhd \mathrm{id})| &= \\
\langle r_0(c'_0 \lhd \mathrm{id})| \dots |r_n(c'_n \lhd \mathrm{id})| &= \\
\langle c'_0 \lhd \mathrm{id}| \dots |c'_n \lhd \mathrm{id}|.
\end{aligned}$$

This can be then proven for any $n \in \mathbb{N}$. □

Corollary 9.15. *Any stochastic process $f \in \mathsf{StochProc}(\mathbb{X}, \mathbb{Y})$ with a family of conditionals c_n gives rise to the observational sequence*

$$\begin{aligned}
\mathrm{obs}(f) = [\langle (c_n \lhd \mathrm{id})\colon (X_0 \times Y_0 \times \cdots \times X_{n-1} \times Y_{n-1}) \times X_n \to \\
(X_0 \times Y_0 \times \cdots \times X_n \times Y_n) \times Y_n \rangle]_{\approx},
\end{aligned}$$

which is independent of the chosen family of conditionals.

Proof. Any two families of conditionals for f give rise to the same n-stage processes in Stoch (by Proposition 9.14). Being a productive category, observational sequences are determined by their n-stage procesess. □

9.3. **Stochastic streams are stochastic processes.** Stochastic monoidal streams and stochastic processes not only are the same thing but they compose in the same way: they are *isomorphic* as categories.

Proposition 9.16. *An observational sequence in* Stoch*,*

$$[\langle g_n \colon M_{n-1} \otimes X_n \to M_n \otimes Y_n \rangle]_{\approx} \in \mathbf{Obs}(\mathbb{X}, \mathbb{Y})$$

gives rise to a stochastic process $\mathrm{proc}(g) \in \mathsf{StochProc}(\mathbb{X}, \mathbb{Y})$ *defined by* $\mathrm{proc}(g)_n = \underline{g_0} \fatsemi \underline{g_1} \fatsemi \dots \fatsemi \underline{g_n} \fatsemi \underline{\varepsilon_{M_n}}$.

Proof. The symmetric monoidal category Stoch is productive: by Lemma 6.15, observational sequences are determined by their n-stage truncations

$$\langle g_0 | \dots | g_n | \in \mathbf{Stage}_n(\mathbb{X}, \mathbb{Y}).$$

Each n-stage truncation gives rise to the n-th component of the stochastic process, $\mathrm{proc}(g)_n = \underline{g_0} \fatsemi \underline{g_1} \fatsemi \dots \fatsemi \underline{g_n} \fatsemi \underline{\varepsilon_{M_n}}$, and this is well-defined: composing the morphisms is invariant to *sliding equivalence*, and the last discarding map is natural.

It only remains to show that they satisfy the marginalisation property. Indeed,

$$\begin{aligned} \mathrm{proc}(g)_{n+1} \fatsemi \underline{\varepsilon_{n+1}} &= \underline{g_0} \fatsemi \underline{g_1} \fatsemi \dots \fatsemi \underline{g_{n+1}} \fatsemi \underline{\varepsilon_{M_{n+1}}} \fatsemi \underline{\varepsilon_{Y_{n+1}}} \\ &= \underline{g_0} \fatsemi \underline{g_1} \fatsemi \dots \fatsemi \underline{g_n} \fatsemi \underline{\varepsilon_{M_n}} \\ &= \mathrm{proc}(g)_n. \end{aligned}$$

Thus, $\mathrm{proc}(g)$ is a stochastic process in $\mathsf{StochProc}(\mathbb{X}, \mathbb{Y})$. □

Proposition 9.17. *Let* $f \in \mathsf{StochProc}(\mathbb{X}, \mathbb{Y})$*, we have that* $\mathrm{proc}(\mathrm{obs}(f)) = f$.

Proof. Indeed, for c_n some family of conditionals,

$$f_n = c_0 \lhd \dots \lhd c_n = (c_0 \lhd \mathrm{id}) \fatsemi \dots \fatsemi (c_n \lhd \mathrm{id}) \fatsemi \varepsilon_{M_n}.$$ □

Theorem 9.18. *Observational sequences in* Stoch *are in bijection with stochastic processes.*

Proof. The function obs is injective by Proposition 9.17. We only need to show it is also surjective.

We will prove that any n-stage process $\langle g_0 | \dots | g_n |$ can be equivalently written in the form $\langle (c_0 \lhd \mathrm{id}) | \dots | (c_n \lhd \mathrm{id}) |$. We proceed by induction. Given any $\langle g_0 |$ we use conditionals and dinaturality to rewrite it as

$$\langle g_0 | = \langle c_0 \lhd c_M | = \langle c_0 \lhd \mathrm{id} | .$$

Given any $\langle c_0 \lhd \mathrm{id} | \dots | c_n \lhd \mathrm{id} | g_{n+1} |$, we use again conditionals and dinaturality to rewrite it as

$$\begin{aligned} \langle c_0 \lhd \mathrm{id} | \dots | c_n \lhd \mathrm{id} | g_{n+1} | = \\ \langle c_0 \lhd \mathrm{id} | \dots | c_n \lhd \mathrm{id} | c_{n+1} \lhd c_M | = \\ \langle c_0 \lhd \mathrm{id} | \dots | c_n \lhd \mathrm{id} | c_{n+1} \lhd \mathrm{id} | . \end{aligned}$$

We have shown that obs is both injective and surjective. □

Theorem 9.19 (From Theorem 9.19)**.** *The category* StochProc *of stochastic processes is monoidally isomorphic to the category* Stream *over* Stoch*.*

Proof. We have shown in Theorem 9.18 that proc is a bijection. Let us show that it preserves compositions. Indeed,

$$\begin{aligned}\mathrm{proc}(g \fatsemi h)_n &= \underline{g_0} \fatsemi \underline{h_0} \fatsemi \ldots \fatsemi \underline{g_n} \fatsemi \underline{h_n} \fatsemi \underline{\varepsilon_{M_n \otimes N_n}} \\ &= \underline{g_0} \fatsemi \ldots \fatsemi \underline{g_n} \fatsemi \underline{h_0} \fatsemi \ldots \fatsemi \underline{h_n} \fatsemi \underline{(\varepsilon_{M_n} \otimes \varepsilon_{N_n})} \\ &= \underline{g_0 \ldots g_n} \fatsemi \underline{\varepsilon_{M_n}} \fatsemi \underline{h_0 \ldots h_n} \fatsemi \underline{\varepsilon_{N_n}} \\ &= \mathrm{proc}(g)_n \fatsemi \mathrm{proc}(h)_n.\end{aligned}$$

It also trivially preserves the identity. It induces thus an identity-on-objects functor which is moreover an equivalence of categories. Let us finally show that it preserves tensoring of morphisms.

$$\begin{aligned}\mathrm{proc}(g \otimes h)_n &= \underline{(g_0 \otimes h_0)} \fatsemi \ldots \fatsemi \underline{(g_n \otimes h_n)} \fatsemi \underline{\varepsilon_{M_n \otimes N_n}} \\ &= \underline{(g_0 \otimes h_0)} \fatsemi \ldots \underline{((g_n \varepsilon_{M_n}) \otimes (h_n \varepsilon_{N_n}))} \\ &= \underline{(g_0 \ldots g_n \varepsilon_{M_n})} \otimes \underline{(h_0 \ldots h_n \varepsilon_{N_n})} \\ &= \mathrm{proc}(g)_n \otimes \mathrm{proc}(h)_n.\end{aligned}$$

It is thus also a monoidal equivalence. □

We expect that the theorem above can be generalised to interesting categories of probabilistic channels over measurable spaces (such as the ones covered in [Pan99, CJ19, Fri20]).

Corollary 9.20. StochProc *is a feedback monoidal category.*

9.4. **Examples.** We have characterized in two equivalent ways the notion of controlled stochastic process. This yields a categorical semantics for probabilistic dataflow programming: we may use the syntax of feedback monoidal categories to specify simple stochastic programs and evaluate their semantics in StochProc.

Example 9.21 (Random Walk)**.** Recall the morphism $\mathsf{walk} \in \mathsf{Stream}(\mathbf{1}, \mathbb{Z})$ that we depicted back in Figure 4.

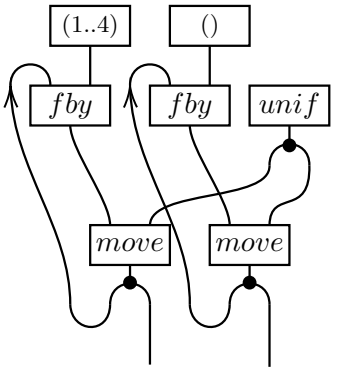
Here, $\mathsf{unif} \in \mathsf{Stream}(\mathbf{1}, \{-1, 1\})$, is a uniform random generator that, at each step, outputs either 1 or (-1). The output of this uniform random generator is then added to the current position, and we declare the starting position to be 0. Our implementation of this morphism [Rom22], following the definitions from Section 7 is, by Theorem 9.19, a discrete stochastic process, and it produces samples like the following ones.

$$[0, 1, 0, -1, -2, -1, -2, -3, -2, -3, \ldots]$$
$$[0, 1, 2, 1, 2, 1, 2, 3, 4, 5, \ldots]$$
$$[0, -1, -2, -1, -2, -1, 0, -1, 0, -1, \ldots]$$

Example 9.22 (Ehrenfest model)**.** The Ehrenfest model [Kel11, §1.4] is a simplified model of particle diffusion.

Assume we have two urns with 4 balls, labelled from 1 to 4. Initially, the balls are all in the first urn. We randomly (and uniformly) pick an integer from 1 to 4, and the ball labelled by that number is removed from its box and placed in the other box. We iterate the procedure, with independent uniform selections each time.

Our implementation of this morphism [Rom22], following the definitions from Section 7 yields samples such as the following.



$$
\begin{aligned}
&\mathsf{ehr} = \\
&(1...4) \otimes (); \mathbf{fbk}(\sigma; \\
&\mathsf{fby} \otimes \mathsf{fby} \otimes \mathsf{unif}; \\
&\mathrm{id} \otimes \mathrm{id} \otimes \mathsf{copy}; \sigma; \\
&\mathsf{move} \otimes \mathsf{move}; \\
&\mathsf{copy})
\end{aligned}
$$

FIGURE 35. Ehrenfest model: sig. flow graph and morphism.

$$
\begin{aligned}
&[([2,3,4],[1]), \quad ([3,4],[1,2]), \quad ([1,3,4],[2]),\\
&([1,4],[2,3]), \quad ([1],[2,3,4]), \quad ([],[1,2,3,4]),\\
&([2],[1,3,4]), \qquad\qquad \dots]
\end{aligned}
$$

10. Conclusions

Monoidal streams are a common generalization of streams, causal functions and stochastic processes. In the same way that streams give semantics to dataflow programming [WA$^+$85, HLR92] with plain functions, monoidal streams give semantics to dataflow programming with monoidal theories of processes. Signal flow graphs are a common tool to describe control flow in dataflow programming. Signal flow graphs are also the natural string diagrams of feedback monoidal categories. Monoidal streams form a feedback monoidal category, and signal flow graphs are a formal syntax to describe and reason about monoidal streams. The second syntax we present comes from the type theory of monoidal categories, and it is inspired by the original syntax of dataflow programming. We have specifically studied stochastic dataflow programming, but the same framework allows for *linear*, *quantum* and *effectful* theories of resources.

The literature on dataflow and feedback is rich enough to provide multiple diverging definitions and approaches. What we can bring to this discussion are universal constructions. Universal constructions justify some mathematical object as *the canonical object* satisfying some properties. In our case, these exact properties are extracted from three, arguably underappreciated, but standard category-theoretic tools: *dinaturality*, *feedback*, and *coalgebra*. *Dinaturality*, profunctors and coends, sometimes regarded as highly theoretical developments, are the natural language to describe how processes communicate and compose. *Feedback*, sometimes eclipsed by trace in the mathematical literature, keeps appearing in multiple variants across computer science. *Coalgebra* is the established tool to specify and reason about stateful systems.

10.1. **Further work.**

Other theories. Many interesting examples of theories of processes are not monoidal but just *premonoidal categories* [Jef97, Pow02]. For instance, the kleisli categories of arbitrary monads, where effects (e.g. reading and writing to a global state) do not need to commute. Premonoidal streams can be constructed by restricting dinaturality to their *centres*, and they seem to be easily implementable in existing software for category theory [dFTC20]. Another important source of theories of processes that we have not covered is that of *linearly distributive* and **-autonomous categories* [See87, CS97, Blu93, BCST96].

Within monoidal categories, we would like to make monoidal streams explicit in the cases of partial maps [CL02] for dataflow programming with different clocks [UV08], non-deterministic maps [BŞ01, LM09] and quantum processes [CdVP21]. Generalizing this, a notion of process similar to the one we employ is described by Krstic, Launchbury, and Pavlovic [KLP01].

The 2-categorical view. We describe the morphisms of a category as a final coalgebra. However, it is also straightforward to describe the 2-endofunctor that should give rise to this category as a final coalgebra itself.

Implementation of the type theory. Justifying that the output of monoidal streams is the expected one requires some computations, which we have already implemented separately in the Haskell programming language (see [Rom22]).

Agda has similar foundations and supports the coinductive definitions of this text (Section 7). It is possible to implement a whole interpreter for a Lucid-like stochastic programming language with a dedicated parser, but that requires some software engineering effort that we postpone for further work.

Expressivity. A final question we do not pursue here is *expressivity*: the class of functions a monoidal stream can define. Analogous to Bucciarelli and Leperchey's finitary-PCF-interdefinability-classes [BL04] there would be a notion of interdefinability relative to a PROP of generators, even if it remains unclear to us if their hypergraph approach could be transferred to our case.

We are reasonably confident that other questions about expressivity of monoidal streams can be answered from our Theorem 5.9 and the previous literature on the subject. There might be analogous results to those of Kahn, Plotkin and Park [KM77, KP93, Par81] on recursive stream equations and to those of Panangaden and Stark [PS88] on total-monotone-continuous relations. Our streams are synchronous [Mil80, Mil83] and thus it should be of interest to compare monoidal streams over Scott-continuous functions with the "synchronous Kahn networks" of Caspi and Pouzet [CP96].

Acknowledgements. We thank Paweł Sobociński, Edward Morehouse, Niels Voorneveld, George Kaye, Chad Nester, Nathanael Arkor, Prakash Panangaden and Ichiro Hasuo for discussion. We thank the anonymous reviewers at LMCS and LiCS'22, whose comments have helped improve this manuscript multiple times; we thank the reviewers for the earlier versions of this work at NWPT'21 and SYCO8.

References

[AC09] Samson Abramsky and Bob Coecke. Categorical quantum mechanics. *Handbook of quantum logic and quantum structures*, 2:261–325, 2009. arXiv:0808.1023.

[Ack79] William B Ackerman. Data flow languages. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 1087–1095. IEEE, 1979.

[Ada74] Jiří Adamek. Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*, 015(4):589–602, 1974. URL: http://eudml.org/doc/16649.

[Adá05] Jiří Adámek. Introduction to coalgebra. *Theory and Applications of Categories*, 14(8):157–199, 2005.

[Ada19] Jiří Adamek. On Terminal Coalgebras Derived from Initial Algebras. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science, CALCO 2019, June 3-6, 2019, London, United Kingdom*, volume 139 of *LIPIcs*, pages 12:1–12:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CALCO.2019.12.

[AI15] Martín Abadi and Michael Isard. Timely Dataflow: A Model. In Susanne Graf and Mahesh Viswanathan, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 35th IFIP WG 6.1 International Conference, FORTE 2015, Held as Part of the 10th International Federated Conference on Distributed Computing Techniques, DisCoTec 2015, Grenoble, France, June 2-4, 2015, Proceedings*, volume 9039 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2015. doi:10.1007/978-3-319-19195-9_9.

[AM89] Peter Aczel and Nax Mendler. A final coalgebra theorem. In *Category theory and computer science*, pages 357–365. Springer, 1989.

[AW77] Edward A. Ashcroft and William W. Wadge. Lucid, a nonprocedural language with iteration. *Communications of the ACM*, 20(7):519–526, 1977.

[BCLH93] Albert Benveniste, Paul Caspi, Paul Le Guernic, and Nicolas Halbwachs. Data-flow synchronous languages. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium, Noordwijkerhout, The Netherlands, June 1-4, 1993, Proceedings*, volume 803 of *Lecture Notes in Computer Science*, pages 1–45. Springer, 1993. doi:10.1007/3-540-58043-3_16.

[BCST96] Richard F Blute, J Robin B Cockett, Robert AG Seely, and Todd H Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113(3):229–296, 1996.

[BÉ93] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories - The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1993. doi:10.1007/978-3-642-78034-9.

[Bec69] Jon Beck. Distributive laws. In *Seminar on triples and categorical homology theory*, pages 119–140. Springer, 1969.

[BFP16] John C. Baez, Brendan Fong, and Blake S. Pollard. A Compositional Framework for Markov Processes. *Journal of Mathematical Physics*, 57(3):033301, March 2016. arXiv:1508.06448, doi:10.1063/1.4941578.

[BHP+19] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. *Proc. ACM Program. Lang.*, 3(POPL):25:1–25:28, 2019. doi:10.1145/3290338.

[BL04] Antonio Bucciarelli and Benjamin Leperchey. Hypergraphs and degrees of parallelism: A completeness result. In *International Conference on Foundations of Software Science and Computation Structures*, pages 58–71. Springer, 2004.

[Blu93] Richard Blute. Linear logic, coherence, and dinaturality. *Theor. Comput. Sci.*, 115(1):3–41, 1993. doi:10.1016/0304-3975(93)90053-V.

[BM89] Bard Bloom and Albert R Meyer. A remark on bisimulation between probabilistic processes. In *International Symposium on Logical Foundations of Computer Science*, pages 26–40. Springer, 1989.

[BMSS11] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: Step-indexing in the topos of trees. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 55–64. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.16.

[BŞ01] Manfred Broy and Gheorghe Ştefănescu. The algebra of stream processing functions. *Theoretical Computer Science*, 258(1-2):99–129, 2001.

[BSS18] Filippo Bonchi, Jens Seeber, and Paweł Sobociński. Graphical conjunctive queries. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 13:1–13:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.13.

[BSZ14] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In *International Conference on Concurrency Theory*, pages 435–450. Springer, 2014.

[BSZ15] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Full abstraction for signal flow graphs. *ACM SIGPLAN Notices*, 50(1):515–526, 2015.

[CdVP21] Titouan Carette, Marc de Visme, and Simon Perdrix. Graphical language with delayed trace: Picturing quantum computing with finite memory. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470553.

[CFS16] Bob Coecke, Tobias Fritz, and Robert W. Spekkens. A mathematical theory of resources. *Inf. Comput.*, 250:59–86, 2016. doi:10.1016/j.ic.2016.02.008.

[CGH12] J. Robin B. Cockett, Xiuzhan Guo, and Pieter Hofstra. Range Categories I: General theory. *Theory and Applications of Categories*, 26(17):412–452, 2012.

[CJ19] Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. arXiv:1709.00322, doi:10.1017/S0960129518000488.

[CL02] J. Robin B. Cockett and Stephen Lack. Restriction categories I: categories of partial maps. *Theoretical Computer Science*, 270(1-2):223–259, 2002. doi:10.1016/S0304-3975(00)00382-0.

[Cou19] Patrick Cousot. Syntactic and semantic soundness of structural dataflow analysis. In Bor-Yuh Evan Chang, editor, *Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings*, volume 11822 of *Lecture Notes in Computer Science*, pages 96–117. Springer, 2019. doi:10.1007/978-3-030-32304-2_6.

[CP96] Paul Caspi and Marc Pouzet. Synchronous Kahn networks. *ACM SIGPLAN Notices*, 31(6):226–238, 1996.

[CS97] J Robin B Cockett and Robert AG Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997.

[Del19] Antonin Delpeuch. A complete language for faceted dataflow programs. In John Baez and Bob Coecke, editors, *Proceedings Applied Category Theory 2019, ACT 2019, University of Oxford, UK, 15-19 July 2019*, volume 323 of *EPTCS*, pages 1–14, 2019. doi:10.4204/EPTCS.323.1.

[Den74] Jack B. Dennis. First version of a data flow procedure language. In B. Robinet, editor, *Programming Symposium*, pages 362–376. Springer Berlin Heidelberg, 1974. doi:10.1007/3-540-06859-7_145.

[DFL72] J. B. Dennis, J. B. Fosseen, and J. P. Linderman. Data flow schemas. In Andrei Ershov and Valery A. Nepomniaschy, editors, *International Symposium on Theoretical Programming*, pages 187–216, Berlin, Heidelberg, 1972. Springer Berlin Heidelberg. doi:10.1007/3-540-06720-5_15.

[dFTC20] Giovanni de Felice, Alexis Toumi, and Bob Coecke. DisCoPy: Monoidal categories in Python. In *Proceedings of Applied Category Theory, ACT 2019, University of Oxford, UK, 15-19 July 2019*, volume 323 of *EPTCS*, pages 183–197, 2020. doi:10.4204/EPTCS.333.13.

[DLGR+21] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Paweł Sobociński. A canonical algebra of open transition systems. In Gwen Salaün and Anton Wijs, editors, *Formal Aspects of Component Software*, pages 63–81, Cham, 2021. Springer International Publishing.

[DLGR+23] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Paweł Sobociński. Span(Graph): a Canonical Feedback Algebra of Open Transition Systems. *Software and Systems Modeling*, 22:495–520, 2023. arXiv:2010.10069, doi:10.1007/s10270-023-01092-7.

[Fos72] John Blake Fosseen. Representation of algorithms by maximally parallel schemata. Master's thesis, Massachusetts Institute of Technology, 1972.

[Fox76] Thomas Fox. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7):665–667, 1976.

[FR75] Wendell Helms Fleming and Raymond W. Rishel. *Deterministic and Stochastic Optimal Control*. Number vol 1 in Applications of Mathematics. Springer-Verlag, Berlin ; New York, 1975.

[Fri20] Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. URL: http://arxiv.org/abs/1908.07021, arXiv:1908.07021.

[FS19] Brendan Fong and David I Spivak. Supplying bells and whistles in symmetric monoidal categories. *arXiv preprint arXiv:1908.02633*, 2019.

[Gar21] Richard Garner. Stream processors and comodels. *arXiv preprint arXiv:2106.05473*, 2021.

[GKS22] Dan R. Ghica, George Kaye, and David Sprunger. Full abstraction for digital circuits, 2022. arXiv:2201.10456.

[GN03] Simon J. Gay and Rajagopal Nagarajan. Intensional and extensional semantics of dataflow programs. *Formal Aspects Comput.*, 15(4):299–318, 2003. doi:10.1007/s00165-003-0018-1.

[HJ98] Claudio Hermida and Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. Comput.*, 145(2):107–152, 1998. doi:10.1006/inco.1998.2725.

[HLR92] Nicolas Halbwachs, Fabienne Lagnier, and Christophe Ratel. Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE. *IEEE Trans. Software Eng.*, 18(9):785–793, 1992. doi:10.1109/32.159839.

[HMH14] Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 52:1–52:10. ACM, 2014. doi:10.1145/2603088.2603124.

[Hon81] Furio Honsell. *Modelli della teoria degli insiemi, principi di regolarità e di libera costruzione*. PhD thesis, Tesi di Laurea, Università di Pisa, 1981.

[HPW98] Thomas Hildebrandt, Prakash Panangaden, and Glynn Winskel. A relational model of non-deterministic dataflow. In *International Conference on Concurrency Theory*, pages 613–628. Springer, 1998.

[Jac16] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. doi:10.1017/CBO9781316823187.

[Jef97] Alan Jeffrey. Premonoidal categories and flow graphs. *Electron. Notes Theor. Comput. Sci.*, 10:51, 1997. doi:10.1016/S1571-0661(05)80688-7.

[JS91] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in mathematics*, 88(1):55–112, 1991.

[JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447 – 468, 04 1996. doi:10.1017/S0305004100074338.

[Kah74] Gilles Kahn. The semantics of a simple language for parallel programming. *Information processing*, 74:471–475, 1974.

[Kel11] Frank P. Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[KLP01] Sava Krstic, John Launchbury, and Dusko Pavlovic. Categories of processes enriched in final coalgebras. In Furio Honsell and Marino Miculan, editors, *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2030 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2001. doi:10.1007/3-540-45315-6_20.

[KM77] Gilles Kahn and David B. MacQueen. Coroutines and networks of parallel processes. In Bruce Gilchrist, editor, *Information Processing, Proceedings of the 7th IFIP Congress 1977, Toronto, Canada, August 8-12, 1977*, pages 993–998. North-Holland, 1977.

[Kos73] Paul R. Kosinski. A data flow language for operating systems programming. In *Proceeding of ACM SIGPLAN-SIGOPS interface meeting on Programming languages-operating systems*, pages 89–94, 1973.

[KP93] Gilles Kahn and Gordon D. Plotkin. Concrete domains. *Theoretical Computer Science*, 121(1-2):187–277, 1993.

[KS17] Dexter Kozen and Alexandra Silva. Practical coinduction. *Mathematical Structures in Computer Science*, 27(7):1132–1152, 2017. doi:10.1017/S0960129515000493.

[KSW97] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115(2):141–178, 1997.

[KSW99] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. On the algebra of feedback and systems with boundary. In *Rendiconti del Seminario Matematico di Palermo*, 1999.

[KSW02] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. Feedback, trace and fixed-point semantics. *RAIRO-Theor. Informatics Appl.*, 36(2):181–194, 2002. doi:10.1051/ita:2002009.

[Lam68] Joachim Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

[Lam86] J. Lambek. Cartesian closed categories and typed λ-calculi. In Guy Cousineau, Pierre-Louis Curien, and Bernard Robinet, editors, *Combinators and Functional Programming Languages*, Lecture Notes in Computer Science, pages 136–175, Berlin, Heidelberg, 1986. Springer. doi:10.1007/3-540-17184-3_44.

[LM09] Edward A. Lee and Eleftherios Matsikoudis. The semantics of dataflow with firing. *From Semantics to Computer Science: Essays in Honour of Gilles Kahn*, pages 71–94, 2009.

[Lor21] Fosco Loregian. *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi:10.1017/9781108778657.

[LS89] Nancy A. Lynch and Eugene W. Stark. A proof of the Kahn principle for input/output automata. *Information and Computation*, 82(1):81–92, 1989.

[Mac78] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1978. doi:10.1007/978-1-4757-4721-8.

[Mam20] Konstantinos Mamouras. Semantic foundations for deterministic dataflow and stream processing. In Peter Müller, editor, *Programming Languages and Systems - 29th European Symposium on Programming, ESOP 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12075 of *Lecture Notes in Computer Science*, pages 394–427. Springer, 2020. doi:10.1007/978-3-030-44914-8_15.

[Mas53] S. J. Mason. Feedback Theory - Some properties of signal flow graphs. *Proceedings of the Institute of Radio Engineers*, 41(9):1144–1156, 1953. doi:10.1109/JRPROC.1953.274449.

[Mel09] Paul-André Mellies. Categorical semantics of linear logic. *Panoramas et syntheses*, 27:15–215, 2009.

[Mil80] Robin Milner. *A calculus of communicating systems*. Springer, 1980.

[Mil83] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical computer science*, 25(3):267–310, 1983.

[ML69] Saunders Mac Lane. One universe as a foundation for category theory. In *Reports of the Midwest Category Seminar III*, pages 192–200, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.

[Oli84] José Nuno Oliveira. *The formal semantics of deterministic dataflow programs*. PhD thesis, University of Manchester, UK, 1984. URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.376586.

[Pan99] Prakash Panangaden. The category of Markov kernels. *Electronic Notes in Theoretical Computer Science*, 22:171–187, January 1999. doi:10.1016/S1571-0661(05)80602-4.

[Par81] David Park. Concurrency and automata on infinite sequences. In *Theoretical computer science*, pages 167–183. Springer, 1981.

[Pow02] John Power. Premonoidal categories as categories with algebraic structure. *Theor. Comput. Sci.*, 278(1-2):303–321, 2002. doi:10.1016/S0304-3975(00)00340-6.

[PS88] Prakash Panangaden and Eugene W. Stark. Computations, residuals, and the power of indeterminacy. In *International Colloquium on Automata, Languages, and Programming*, pages 439–454. Springer, 1988.

[PW99] John Power and Hiroshi Watanabe. Distributivity for a monad and a comonad. In Bart Jacobs and Jan J. M. M. Rutten, editors, *Coalgebraic Methods in Computer Science, CMCS 1999, Amsterdam, The Netherlands, March 20-21, 1999*, volume 19 of *Electronic Notes in Theoretical Computer Science*, page 102. Elsevier, 1999. doi:10.1016/S1571-0661(05)80271-3.

[Rom20] Mario Román. Comb diagrams for discrete-time feedback. *CoRR*, abs/2003.06214, 2020. arXiv:2003.06214.

[Rom22] Mario Román. Arrow Streams for Dataflow Programming. https://github.com/mroman42/arrow-streams, 12 2022. doi:10.5281/zenodo.15978541.

[Ros96] Sheldon M. Ross. *Stochastic processes*, volume 2. John Wiley & Sons, 1996.

[RR88] Edmund Robinson and Giuseppe Rosolini. Categories of partial maps. *Inf. Comput.*, 79(2):95–130, 1988. doi:10.1016/0890-5401(88)90034-X.

[Rut00] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

[San09] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(4):1–41, 2009.

[See87] Robert A.G. Seely. *Linear logic, *-autonomous categories and cofree coalgebras*. Ste. Anne de Bellevue, Quebec: CEGEP John Abbott College, 1987.

[Sha42] Claude E. Shannon. *The Theory and Design of Linear Differential Equation Machines*. Bell Telephone Laboratories, 1942.

[SJ19] David Sprunger and Bart Jacobs. The differential calculus of causal functions. *CoRR*, abs/1904.10611, 2019. URL: http://arxiv.org/abs/1904.10611, arXiv:1904.10611.

[SK19] David Sprunger and Shin-ya Katsumata. Differentiable causal computations via delayed trace. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. doi:10.1109/LICS.2019.8785670.

[SLG94] Viggo Stoltenberg-Hansen, Ingrid Lindström, and Edward R. Griffor. *Mathematical theory of domains*, volume 22 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 1994.

[TR98] Daniele Turi and Jan Rutten. On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[UV05] Tarmo Uustalu and Varmo Vene. The essence of dataflow programming. In Kwangkeun Yi, editor, *Programming Languages and Systems, Third Asian Symposium, APLAS 2005, Tsukuba, Japan, November 2-5, 2005, Proceedings*, volume 3780 of *Lecture Notes in Computer Science*, pages 2–18. Springer, 2005. doi:10.1007/11575467_2.

[UV08] Tarmo Uustalu and Varmo Vene. Comonadic notions of computation. In Jiří Adámek and Clemens Kupke, editors, *Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science, CMCS 2008, Budapest, Hungary, April 4-6, 2008*, volume 203 of *Electronic Notes in Theoretical Computer Science*, pages 263–284. Elsevier, 2008. doi:10.1016/j.entcs.2008.05.029.

[WA+85] William W Wadge, Edward A Ashcroft, et al. *Lucid, the dataflow programming language*, volume 303. Academic Press London, 1985.

[Wen75] Kung-Song Weng. Steam-oriented computation in recursive data flow schemas. Master's thesis, Massachusetts Institute of Technology, 1975.