Effectful Trace Semantics via Effectful Streams

Filippo Bonchi
Università di Pisa

Elena Di Lavore
Università di Pisa
Tallinn University of Technology

Mario Román
University of Oxford
Tallinn University of Technology

ABSTRACT

We introduce effectful streams, a coinductive semantic universe for effectful dataflow programming and traces. In monoidal categories with conditionals and ranges, we show that effectful streams particularize to families of morphisms satisfying a causality condition. Effectful streams allow us to develop notions of trace and bisimulation for effectful Mealy machines; we prove that bisimulation implies effectful trace equivalence.

ACM Reference Format:
Filippo Bonchi, Elena Di Lavore, and Mario Román. 2024. Effectful Trace Semantics via Effectful Streams. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Traces are sequences that record the outputs of a transition system or a state machine along an execution. They constitute a successful and flexible formalism that can be adapted to the different flavours of transition systems (from *non-deterministic* to *stochastic*). Traces are also a semantic universe of interest for dataflow networks; in fact, they help extending Kahn's original model [Kah74] to a compositional semantics for the non-deterministic case [Jon89].

Traces could be an equally fruitful compositional semantics for fully-fledged dataflow programming [WA+85] (also known as *reactive programming* [EH97] or *stream programming*) – the multiple flavours of traces we developed for machines and networks may serve just the same for stream programming. Yet, categorical semantics for reactive programming has been initially restricted to the pure, cartesian case [UV08, KBH12], the non-deterministic case [HPW98], and only recently to some other monoidal cases [CDVP21, DLdFR22] that still lack effects. Realistic dataflow programming needs arbitrary effects but, in order to give it semantics, it may be necessary to develop trace semantics for the multiple flavours of dataflow programming and then compare each one of them to their classical counterpart for state machines and networks.

This manuscript develops a common solution to these problems. We devise a unified compositional stream semantics for effectful transition systems, effectful Mealy machines, and effectful dataflow programs. *Effectful streams* form this semantics, which fulfills the following desiderata.

- *Fully-abstract programming syntax.* Do-notation is a fully-abstract syntax for effectful categories constituting a refined fragment of *call-by-push-value* [Lev22]. Effectful streams are defined in terms of do-notation and coinduction.
- *Coalgebraic characterization.* Effectful streams form a final coalgebra; elements of this final coalgebra can be constructed coinductively, and we can reason with observational equality.
- *Common generalization.* Effectful streams can be constructed over any effectful category [Lev22]. This includes the Kleisli category of any monad [Mog91], any comonad, or any distributive law [PW99].

We show that the generalization is particularly fruitful. The notion of *causal function* plays a key role in theory of streams since its dawn: Raney [Ran58] introduces them in 1958, proves that they compose and that all functions computed by Mealy machines are causal. When one considers different flavours of Mealy machines, such as non-deterministic or probabilistic, one would expect to have analogous results, but it is far from obvious how causality can be defined in the non deterministic or probabilistic setting.

Recall that a function $f\colon X^\omega \to Y^\omega$, here A^ω stands for the set of streams $\{(a_0, a_1, \dots) \mid a_i \in A\}$, is causal if the n-th output y_n only depends on the first n inputs $x_1, \dots, x_n$. We show that this definition of causality can be generalised by means of *conditionals*, a notion of probability theory that have been recently generalised [Fri20] to arbitrary copy-discard categories [CG99, Jac18]. We define *causal processes* as $\mathbb{N}$-indexed family of morphisms $f_n\colon X_0 \otimes \dots \otimes X_n \to Y_0 \otimes \dots \otimes Y_n$ such that there exist morphisms c_n –the conditionals– satisfying the equation expressed by the diagrams in Figure 1.

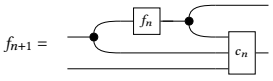


Figure 1: Causality condition for causal processes.

Our main result (Theorem 4.6) provides sufficient conditions guaranteeing that effectful streams are exactly causal processes. The relevance of this results is two fold.

First, effectful streams can be neatly implemented and, at the same time, enjoy a characterisation by means of a universal property. However, they come modulo an equivalence, called *dinaturality*, that is not very helpful in proofs. Instead, the characterisation by means of causal processes is far handier.

Second, the sufficient conditions are general enough to include deterministic, non-deterministic, or stochastic systems. Theorem 4.6 provides a generalization of the results in [Ran58] to these cases and, at the same time, it allows us to easily prove that the semantics provided by effectful streams coincide with the existing notions of trace semantics. We do not need to recast these notions each time we change the flavour of state machine (stochastic, partial, non-deterministic, stateful...).

Conference'17, July 2017, Washington, DC, USA

1.1 Effectful Streams

Effectful streams are a coinductive construction. While induction is widespread and we employ it without a second thought, coinduction [Rut00] is not granted the same trust: every time we employ coinduction, we reassert its validity from basic principles. This can make coinduction look obscure and akward. Instead, we will be reasoning coinductively in the practical style promoted by Kozen and Silva [KS17]: we apply *coinductive hypotheses* that need a *coinductive step* but not a *base case*. This is not to mean our reasoning is not formal: it can be backed by formal principles the same way induction does.

Following this coinductive philosophy, we can say that a stream s is an element (the *head*, s°) followed by a stream (the *tail*, s^+); that constitutes a valid datatype definition. We can then define a constant stream, declaring $\mathbf{1}^\circ = 1$ and $\mathbf{1}^+ = \mathbf{1}$, or even an alternating stream, defining $\boldsymbol{alt}^\circ = 0$, but $\boldsymbol{alt}^{+\circ} = 1$ and $\boldsymbol{alt}^{++} = \boldsymbol{alt}$; we can define the addition of two streams of natural numbers, declaring $(\boldsymbol{x}+\boldsymbol{y})^\circ = \boldsymbol{x}^\circ + \boldsymbol{y}^\circ$ and $(\boldsymbol{x}+\boldsymbol{y})^+ = \boldsymbol{x}^+ + \boldsymbol{y}^+$; or define the stream of the natural numbers, defining $\boldsymbol{nats}^\circ = 0$ and $\boldsymbol{nats}^+ = \mathbf{1} + \boldsymbol{nats}$.

Effectful streams follow exactly the same principle, but with two important differences. The first is that instead of an element of a set, each piece of the stream will represent an *effectful program* – formally, a morphism in an effectful copy-discard category. The second is that each piece of the stream, each program, will not occur in isolation, but it will be allowed to communicate with the next piece via a *memory*. This second principle will allow for causal communication: messages can be passed from the past to the future, but not the other way around.

A first example of an effectful stream is the following counter program that prints the natural numbers by keeping the counting on memory (n): this memory is returned as an output at each step, and received as an input for the next step.

$$\begin{array}{lll} \texttt{counter}()^\circ = & \texttt{counter}(n)^{+\circ} = & \texttt{counter}(n)^{++} = \\ \quad \texttt{print}(0) \rightsquigarrow () & \quad \texttt{print}(n) \rightsquigarrow () & \quad \texttt{counter}(n)^+ \\ \quad \texttt{return}(1) & \quad \texttt{return}(n+1) & \end{array}$$

There is little that we need to assume about the implementation of the function `print`. It can be any effectful process taking a natural number and producing no output, $\texttt{print}\colon N \rightsquigarrow ()$, in an arbitrary category that does not even need to be a monoidal category: this includes Moggi's monadic semantics [Mog91], but so are comonadic [UV08] or arrow semantics [Hug00].

1.2 Example: the stream cipher protocol

Stream cipher protocols encrypt messages of any given length. They are a repeated version of the *one-time pad protocol*: a perfectly secure encryption technique that, however, requires sharing a key ahead of time through a secure channel. In the one-time pad protocol, a first party (e.g. Alice) wants to send a private message, $\boldsymbol{m}$, to a second party (e.g. Bob), through a public channel. Alice and Bob already shared a key – generated randomly by, say, Alice – through a private channel $\boldsymbol{k}$, and this is the ingredient that will now allow secure communication: Alice uses the XOR operation to mix the message and the key, $\boldsymbol{m} \oplus \boldsymbol{k}$, and sends that to Bob; Bob uses now the XOR operation again to mix the received message with the key, $(\boldsymbol{m} \oplus \boldsymbol{k}) \oplus \boldsymbol{k}$. Now, because the XOR operation is a nilpotent algebra, Bob obtains the decrypted message:

$$(\boldsymbol{m} \oplus \boldsymbol{k}) \oplus \boldsymbol{k} = \boldsymbol{m} \oplus (\boldsymbol{k} \oplus \boldsymbol{k}) = \boldsymbol{m} \oplus 0 = \boldsymbol{m}.$$

An attacker listening to the public channel will only receive the encrypted message, $\boldsymbol{m} \oplus \boldsymbol{k}$, which is perfectly uninformative if they do not know the value of the key. That is, the one-time pad protocol is secure, but it still comes with a problem: as soon as the key is used once, it cannot be reused safely; in order to send a long encrypted message, we need an equally long pre-shared key.

The *stream cipher protocol* is a solution to this problem. Instead of using the pre-shared key to encrypt and decrypt, Alice and Bob now use the pre-shared key generated by Alice as the common seed of two private identical random number generators: now they have a inexhaustible source of shared random numbers, and they can use these to repeatedly execute the one-time pad protocol to communicate messages of arbitrary length.

The stream cipher is perfectly secure, but it is necessarily an idealized protocol: it is impossible to create infinite and coupled sources of true randomness; it is only possible to approximate them with pseudorandom number generators. The security of this protocol relies on assuming that the pseudorandom number generator cannot be distinguished from an actual random number generator.

Our signature for the stream cipher contains a single type C. It contains three effects: *(i)* a nullary effect that initializes a common seed for the coupled generators, $\texttt{seed}\colon () \rightsquigarrow ()$; *(ii)* a single-output effect that extracts a random symbol from the first generator, $\texttt{rand}_A\colon () \rightsquigarrow C$; and *(iii)* the same effect for the second generator, $\texttt{rand}_B\colon () \rightsquigarrow C$. It contains a single operator, $(\oplus)\colon C \otimes C \to C$, representing the *xor* operation.

$$\begin{array}{l} \texttt{alice(m)}^\circ = \\ \quad \texttt{seed}() \rightsquigarrow \\ \quad \texttt{rand}_a() \rightsquigarrow k_a \\ \quad \texttt{return}(m \oplus k_a) \\ \texttt{alice(m)}^{+\circ} = \\ \quad \texttt{rand}_a() \rightsquigarrow k_a \\ \quad \texttt{return}(m \oplus k_a) \\ \texttt{alice(m)}^{++} = \texttt{alice(m)}^+ \end{array} \qquad \begin{array}{l} \texttt{bob(m)}^\circ = \\ \quad \texttt{rand}_b() \rightsquigarrow k_b \\ \quad \texttt{return}(m \oplus k_b) \\ \texttt{bob(m)}^+ = \texttt{bob(m)} \end{array}$$

Figure 2: Components of the stream cipher protocol.

The agents participating in the stream cipher protocol are described in Figure 2. Executing $\texttt{alice}\colon C^* \to C^*$ first, copying its output and then executing $\texttt{bob}\colon C^* \to C^*$ over one of the copies produces the following resulting protocol $\texttt{cipher}\colon C^* \to C^* \otimes C^*$. Under the right assumptions, we will be able to prove correctness for this protocol (Section 3.2).

$$\begin{array}{l} \texttt{cipher(m)}^\circ = \\ \quad \texttt{init}() \rightsquigarrow () \\ \quad \texttt{rand}_a() \rightsquigarrow k_a \\ \quad \texttt{rand}_b() \rightsquigarrow k_b \\ \quad \texttt{return}(m \oplus k_a \oplus k_b, m \oplus k_a) \end{array} \qquad \begin{array}{l} \texttt{cipher(m)}^{+\circ} = \\ \quad \texttt{rand}_a() \rightsquigarrow k_a \\ \quad \texttt{rand}_b() \rightsquigarrow k_b \\ \quad \texttt{return}(m \oplus k_a \oplus k_b, m \oplus k_a) \\ \texttt{cipher(m)}^{++} = \texttt{cipher(m)}^+ \end{array}$$

In Section 3.2, we prove that the stream cipher protocol is secure relative to a secure pseudorandom number generator. We follow a recent proof of correctness of the *one-time pad protocol* for an arbitrary nilpotent bialgebra in a symmetric monoidal category due to Broadbent and Karvonen [BK23].

1.3 Contributions

We introduce *effectful streams* (Definition 3.1); a common generalization of classical streams, stateful morphism sequences [SK19], monoidal streams [DLdFR22], that uses premonoidal categories [PT97].

Our notion captures multiple examples that could not be captured neither by the "stateful morphism sequences" of Katsumata and Sprunger [SK19] nor by the "monoidal streams" of Di Lavore, de Felice and Román [DLdFR22]. These include relational streams, partial streams, and partial stochastic streams.

We generalise the notion of causal functions, originally introduced in [Ran58], to *causal processes* (Definition 4.1) by exploiting conditionals in copy-discard categories. Our main result (Theorem 4.6) states that, under the additional assumption of existence of ranges, effectful streams are indeed causal processes.

We illustrate conditionals and ranges for the categories of relations, partial functions and partial sthochastic functions and thus, by means of Theorem 4.6, we obtain characterisations for relational, partial and partial stochastic streams as causal processes (Corollaries 4.11, 4.13, 4.15) .

We exploit such characterisations to prove that the effectful stream semantics of *effectful Mealy machines*, a generalization of Mealy machines that we introduce in Definition 5.1, coincide with existing trace semantics for non-deterministic, partial and probabilistic systems. As a further sanity check, we show two key properties of our semantics: it is compostional (Proposition 5.6) and it is entailed by a certain notion of bisimilarity (Theorem 5.10).

1.4 Related work

Causal dataflow programming. The theory of streams for causal dataflow programming comes from the work of Kahn [Kah74]. The particular coalgebraic approach we employ comes appears briefly in the work of Uustalu and Vene [UV08], and it was developed in more detail by Sprunger and Katsumata [SK19]: they first propose cartesian streams as a categorical modal for dataflow programming. This model was later refined to the monoidal by Di Lavore, de Felice and Román [DLdFR22], although we significantly differ from their design choices: *(i)* by taking the more general, but also more natural and comfortable setting of *effectful categories*, which allows us to reason with programs in *do-notation*; *(ii)* by avoiding a feedback type-theory and feedback diagrams that, although sound, are not complete except on the presence of a coinduction rule; and *(iii)* by instead obtaining the axioms of feedback and delay as derived rules arising from the interaction between *do-notation* and coinduction. Finally, we are able to capture examples like partial, relational and partial stochastic streams, which were out of the reach for monoidal streams. Streams have also been studied in the case of quantum computations [CDVP21] and digital circuits [GKS22]. Causal functions were introduced by Raney [Ran58] and later studied in different aspects [SJ19, GKS22, SK19].

Effectful Mealy machines. Our effectful Mealy machines are a natural extension of the bicategory of "processes" that Katis, Sabadini and Walters defined as the suspension-loop of a base symmetric monoidal category of processes [KSW97]. Most of the categorical literature on Mealy machines has been separated from monoidal categories [CP20, SBBR13]; restricted to the cartesian case [BLLL23]; or expanded Katis, Sabadini and Walters' work [DLGR$^+$23]. Finally, note the term "monoidal automata" is used to indicate automata recognizing monoidal languages [ES22, ES23]. Finite state automata can be given a finite axiomatisation [PZ23], but we are not concerned with this issue in the present manuscript.

Bisimulation and traces. Bisimulation can be defined in general for coalgebras [Rut95, Rut00]. When the base category is cartesian closed, transition systems can be expressed as coalgebras for a functor, which gives a notion of bisimulation for them. Our approach is more general and coincides with the coalgebraic one under reasonable assumptions. Probabilistic and metric bisimulation, in particular, receive a lot of attention [BDEP97, DDLP06, BBLM17, BSS21, BSV22]. Different notions of equality for effectful and monoidal programs have been studied: effectful applicative bisimilarity [DLGL17], equivalence for programs with effects and general recursion [Voo20], and monoidal traces [Jac10].

Feedback. Feedback monoidal categories and their free construction have been defined multiple times in the literature [KSW97, HMH14, BHP$^+$19], but the construction is originally due to Katis, Sabadini and Walters [KSW02]. It was extended to *delayed* feedback in [DLGR$^+$23]. Our running example owes to the description of the one-time pad protocol in symmetric monoidal categories by Broadbent and Karvonen [BK23].

2 PRELIMINARIES

2.1 Effectful Copy-Discard Categories

Effectful categories allow us to talk about pure computations, as morphisms in a monoidal category $\mathbf{V}$, and effectful computations, as morphisms in a premonoidal category $\mathbf{C}$.

Using a triple of categories for the semantics of values, pure and effectful processes is a refinment by Jeffrey [Jef97] of the distinction between values and computations in the work of Levy [Lev22], Power and Thielecke [PT97].

Definition 2.1. An *effectful copy-discard category* is a triple of categories with two identity-on-objects functors, $\mathbf{V} \to \mathbf{P} \to \mathbf{C}$, where

(1) the first category, $\mathbf{V}$, is a cartesian monoidal category representing "values" that can be copied and discarded;
(2) the second category, $\mathbf{P}$, is a monoidal category representing "pure computations" or "pure processes" whose execution can be reordered without altering the result; and
(3) the third category, $\mathbf{C}$, is a premonoidal category representing "effectful computations" or "effectful processes" with a fixed order of execution.

The first identity-on-objects functor, $\mathbf{V} \to \mathbf{C}$, must preserve the monoidal structure strictly; the second identity-on-objects functor, $\mathbf{P} \to \mathbf{C}$, must preserve the premonoidal structure strictly.

Remark 2.2. Setting some of the identity-on-object functors to be identities, we recover multiple notions from the literature. A *Freyd category* [PT97] is an effectful copy-discard category with only values and effectful computations, $\mathbf{V} \to \mathbf{C}$; an *effectful category* [JHH09, Rom22] is an effectful copy-discard category with only pure end effectful computations, $\mathbf{P} \to \mathbf{C}$; a *copy-discard category*

[FL23] with chosen maps, or a *monoidal Freyd category*, is an effectful copy-discard category with values and pure computations, $\mathbf{V} \to \mathbf{P}$.

An effectful copy-discard category is strict whenever the three categories that conform it are strict. A strictification theorem holds for effectful copy-discard categories, where each effectful copy-discard category is equivalent to a strict one and the equivalence preserves the cartesian, monoidal and premonoidal structures. Strict effectful copy-discard categories and effectful copy-discard functors between them form a category, **Ecd**.

Remark 2.3. The Kleisli of any monad $T : \mathbf{Set} \to \mathbf{Set}$, forms an effectful copy-discard category, $(\mathbf{V}, \mathsf{Kl}(\mathcal{Z}(T)), \mathsf{Kl}(T))$, where $\mathcal{Z}(T)$ is the centre of the monad [CLZ23]. More generally, any Freyd category $(\mathbf{V}, \mathbf{C})$ induces an effectful copy-discard category, $(\mathbf{V}, \mathcal{Z}(\mathbf{C}), \mathbf{C})$, where $\mathcal{Z}(\mathbf{C})$ is the centre of the premonoidal category.

2.2 Effectful Copy-Discard Do-Notation

Definition 2.4. An *effectful copy-discard graph*, $(O, \mathcal{V}, \mathcal{P}, C)$, consists of a set of objects, O; a set of value generators, $\mathcal{V}(X_1, ..., X_n; Y)$, for each list of objects, $X_1, ..., X_n, Y \in O$; a set of pure generators, $\mathcal{P}(X_1, ..., X_n; Y_1, ..., Y_m)$, and a set of computation generators, $C(X_1, ..., X_n; Y_1, ..., Y_m)$, for each two lists $X_1, ..., X_n, Y_1, ..., Y_m \in O$.

While computation generators are allowed to have multiple outputs, value generators are only allowed a single output. This is not a real restriction: values form a cartesian category where each morphism, $f : X \to Y_1 \times Y_2$, can be reconstructed from its two projections, $f_1 : X \to Y_1$ and $f_2 : X \to Y_2$; instead of using a single generator for a multiple-output value, we can use a family of generators.

Effectful copy-discard graphs form a category, **EcdGraph**, endowed with a forgetful functor from the category of strict effectful copy-discard categories, $\mathsf{Forget} : \mathbf{Ecd} \to \mathbf{EcdGraph}$ (see Definition B.1).

Definition 2.5. Do-notation for effectful copy-discard categories, over a given effectful copy-discard graph $(O, \mathcal{V}, \mathcal{P}, \mathcal{E})$ is the type theory generated by the following rules, quotiented by α-equivalence. It contains two types of derivations, one derives simple values ($\vdash$), the other derives pure and effectful computations, ($\Vdash$).

$$\textsc{Return}\ \frac{\Gamma \vdash t_1 : X_1 \quad ... \quad \Gamma \vdash t_n : X_n}{\Gamma \Vdash \mathsf{return}(t_1, ..., t_n) : X_1, ..., X_n} \qquad \textsc{Variable}\ \frac{}{(x_i : X_i) \in \Gamma \vdash x_i : X_i}$$

$$\textsc{Value Generator}\ \frac{\Gamma \vdash t_1 : X_1 \quad ... \quad \Gamma \vdash t_n : X_n}{\Gamma \vdash f(t_1, ..., t_n) : Y}$$

$$\textsc{Pure Generator}\ \frac{y_1 : Y_1, ..., y_m : Y_m, \Gamma \Vdash p : Z_1, ..., Z_m \quad \Gamma \vdash t_1 : X_1, ..., \Gamma \vdash t_n : X_n}{\Gamma \Vdash g(t_1, ..., t_n) \to y_1, ..., y_m \fatsemi p : Z_1, ..., Z_m}$$

$$\textsc{Effectful Generator}\ \frac{y_1 : Y_1, ..., y_m : Y_m, \Gamma \Vdash p : Z_1, ..., Z_m \quad \Gamma \vdash t_1 : X_1, ..., \Gamma \vdash t_n : X_n}{\Gamma \Vdash h(t_1, ..., t_n) \rightsquigarrow y_1, ..., y_m \fatsemi p : Z_1, ..., Z_m}$$

There exists an instance of the Value rule for each value generator, $f \in \mathcal{V}(X_1, ..., X_n; Y)$; there exists an instance of the Pure rule for each pure generator, $g \in \mathcal{P}(X_1, ..., X_n; Y_1, ..., Y_m)$; there exists an instance of the Effectful rule for each effectful generator of the form $h \in \mathcal{E}(X_1, ..., X_n; Y_1, ..., Y_m)$.

Remark 2.6. Reading the notation is considerably easier when we remove the statement separator, ($\fatsemi$), and replace it with a line break. We do so for the rest of the text.

Remark 2.7 (Interchange quotienting). Consider the following two derivations, whose only difference is the order of execution of a pure statements over distinct variables (meaning, in the example, that no x_i appears in any c_j). Then, we can interchange these two statements without affecting the meaning of the program.

$$\left.\begin{matrix} f(a_1, ..., a_n) \to x_1, ..., x_m \\ h(c_1, ..., c_p) \rightsquigarrow y_1, ..., y_q \end{matrix}\right\} = \left\{\begin{matrix} h(c_1, ..., c_p) \rightsquigarrow y_1, ..., y_q \\ f(a_1, ..., a_n) \to x_1, ..., x_m \end{matrix}\right.$$

Following the same principle, we also allow two pure statements to interchange.

$$\left.\begin{matrix} f(a_1, ..., a_n) \to x_1, ..., x_m \\ g(b_1, ..., b_p) \to y_1, ..., y_q \end{matrix}\right\} = \left\{\begin{matrix} g(b_1, ..., b_p) \to y_1, ..., y_q \\ f(a_1, ..., a_n) \to x_1, ..., x_m \end{matrix}\right.$$

Proposition 2.8. *Do-notation derivations over an effectful copy-discard graph, $\mathcal{G} = (O, \mathcal{V}, \mathcal{P}, \mathcal{E})$, quotiented by interchange, form an effectful copy-discard category,* $\mathsf{Do}_{\mathcal{G}}(\mathcal{V}) \to \mathsf{Do}(\mathcal{P}) \to \mathsf{Do}(\mathcal{E})$.

Proof. See Appendix, Proposition B.5. □

Proposition 2.9. *Do-notation derivations over an effectful copy-discard graph quotiented by interchange form the free effectful copy-discard category, over that graph. That is, there exists an adjunction,* $\mathsf{Do} \dashv \mathsf{Forget}$.

Example 2.10. The category **Cipher** is the syntactic category on which we wrote our stream cipher protocol (Section 1.2). We define it to be the free effectful copy-discard category over the effectful copy-discard graph containing only the following non-empty sets: the set of objects $O = \{C\}$, values $V(C, C; C) = \{\oplus\}$, pure generators $\mathcal{P}(; C) = \{\mathtt{unif}\}$, and effectful generators, $\mathcal{E}(;) = \{\mathtt{seed}\}$ and $\mathcal{E}(; C) = \{\mathtt{rand}_a, \mathtt{rand}_b\}$. There is a single generator that has not been yet used, `unif`; it will appear later, in Section 3.2.

2.3 Copy-discard categories

Copy-discard categories are monoidal categories in which objects can be copied and discarded, not necessarily in a natural way, and are also known as gs-monoidal categories [CG99]. All Kleisli categories of monads on the category **Set** of sets and functions have a copy-discard structure inherited from **Set**. A copy-discard structure is a *copy* morphism $\multimap\!\!\subset : A \to A \otimes A$ and a *discard* morphism $\multimap\!\!\bullet : A \to I$, for all objects A.

Remark 2.11. Only values can be copied and discarded. In general, executing f once and copying its output is not be the same as copying its input and then executing f twice; similarly, discarding the output of f is not the same as never executing it. Morphisms that can be copied are called *deterministic*; morphisms that can be discarded are called *total* (Figure 3).

Total morphisms of a copy-discard category $(\mathbf{V}, \mathbf{C})$ form a wide subcategory $\mathsf{Tot}(\mathbf{C})$ that we will commonly choose as the pure morphisms of a full effectful copy-discard category.

$$\left.\begin{array}{l} f(x) \to y_1 \\ f(x) \to y_2 \\ \texttt{return}(y_1, y_2) \end{array}\right\} = \left.\begin{array}{l} f(x) \to y \\ \texttt{return}(y, y) \end{array}\right\}; \quad \texttt{return}() = \left.\begin{array}{l} f(x) \to y \\ \texttt{return}() \end{array}\right\};$$

Figure 3: Deterministic and total morphisms.

Conditionals are a way of splitting morphisms $f\colon X \to A \otimes B$ with two outputs in a way that produces the first output, A, first and then the second output, B. Quasi-totality ensures that, whenever the output A is produced, so is the output B. Conditionals were introduced by Fritz [Fri20] for Markov categories, copy-discard categories where all morphisms are total. Their extension to generic copy-discard categories requires the introduction of the quasi-totality condition to preserve some of the properties of conditionals in Markov categories [DLR23].

Definition 2.12. A copy-discard category $(\mathbf{V}, \mathbf{P})$ has *conditionals* if, for every morphism $f\colon X \to A \otimes B$, there exist non-necessarily unique morphisms $m\colon X \to A$ and $c\colon A \otimes X \to B$, called *marginal* and *conditional*, such that the following equation holds.

$$\left.\begin{array}{l} f(x) \to a, b \\ \texttt{return}(a, b) \end{array}\right\} = \left.\begin{array}{l} m(x) \to a \\ c(x, a) \to b \\ \texttt{return}(a, b) \end{array}\right\}$$

We say that f has a *quasi-total conditional* whenever it additionally satisfies the following equation.

$$\left.\begin{array}{l} c(a, x) \to b_1 \\ c(a, x) \to b_2 \\ \texttt{return}() \end{array}\right\} = \left.\begin{array}{l} c(x, a) \to b \\ \texttt{return}() \end{array}\right\}$$

String diagrams for these conditions can be found in the Appendix, Figure 8.

Section 4 makes use of both conditionals and quasi-total conditionals to characterise streams as causal processes. When a conditional is quasi-total, $f \fatsemi (\mathrm{id}_A \otimes \multimap_B)$ is always a marginal for it [DLR23, Proposition 3.14].

Remark 2.13. Conditionals are not unique and they cannot be, otherwise the category structure collapses to a poset [Fri20, DLR23]. However, all conditionals with respect to the same marginal should behave in the same way on all the inputs that can be produced by that marginal. This intuition is captured by the definition of *ranges*.

Ranges were introduced in [DLdFR22] for similar purposes. Here, we take a slightly more general definition to take into account that some maps are not total.

Definition 2.14. A copy-discard category, $(\mathbf{V}, \mathbf{P})$, has *ranges* if, for every morphism $m\colon X \to A$, there exists an object, the *range* $R \in \mathbf{C}_{obj}$, a deterministic morphism $r\colon A \otimes X \to A \otimes X$ and a total morphism $i\colon R \to A \otimes X$ such that *(i)* the range does not modify the output of the morphism,

$$\left.\begin{array}{l} m(x) \to a \\ \texttt{return}\ (x, a) \end{array}\right\} = \left.\begin{array}{l} m(x) \to a \\ r(x, a) \to (x', a') \\ \texttt{return}\ (x', a') \end{array}\right\};$$

and *(ii)* any two morphisms $c, d\colon A \otimes X \otimes Y \to B$ that equalize the whole range,

$$\left.\begin{array}{l} m(x) \to a \\ c(x, a, y) \to b \\ \texttt{return}\ (a, b) \end{array}\right\} = \left.\begin{array}{l} m(x) \to a \\ d(x, a, y) \to b \\ \texttt{return}\ (a, b) \end{array}\right\}$$

do also equalize its total part,

$$\left.\begin{array}{l} \iota(s) \to (x, a) \\ c(x, a, y) \to b \\ \texttt{return}\ (a, b) \end{array}\right\} = \left.\begin{array}{l} \iota(s) \to (x, a) \\ d(x, a, y) \to b \\ \texttt{return}\ (a, b) \end{array}\right\}$$

String diagrams for these conditions are provided in the Appendix, Figure 9.

2.4 Dinaturality and Profunctors

Effectful streams will be constructed by glueing together multiple processes. Processes can be collected into profuctors: sets, $P(X; Y)$, indexed contravariantly by a category of input types, $X \in \mathbb{A}$, and covariantly by a category of output types, $Y \in \mathbb{B}$. The input category acts contravariantly with an action $(\succ)\colon \mathbb{A}(X'; X) \times P(X; Y) \to P(X'; Y)$, while the output category acts covariantly with an action $(\prec)\colon P(X; Y) \times \mathbb{B}(Y; Y') \to P(X; Y')$. Both actions must be compatible and satisfy certain axioms that amount to say that a profunctor from $\mathbb{A}$ to $\mathbb{B}$ is the same thing as a functor $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$.

Glueing two profunctors, $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$ and $Q\colon \mathbb{B}^{op} \times \mathbf{C} \to \mathbf{Set}$, along a common category (with different variance) starts by considering the set of pairs of processes, $(P \times Q)(X, Y_1, Y_2, Z) = P(X; Y_1) \times Q(Y_2; Z)$. We would like to impose that the type of the first output, Y_1, coincides with the type of the second input Y_2, but doing so naively would introduce redundancy: for each morphism $r\colon Y_1 \to Y_2$ and any pair of processes $p \in P(X; Y_1)$ and $q \in Q(Y_2; Z)$, we may consider either the tuple where r acts into the first component, $(p \prec r, q)$ or the tuple where r acts into the second component, $(p, r \succ q)$. Even when these are two different types of processes, and except for the instant in which they execute r, they represent the same process – they are *dinaturally equal*.

Definition 2.15. Dinaturality, $(\sim)$, is the minimal equivalence relation that equalizes the contravariant and covariant actions on a family of sets indexed covariantly and contravariantly by the same category, $S\colon \mathbb{B} \times \mathbb{B}^{op} \to \mathbf{Set}$. Explicitly, on the set $\sum_{Y \in \mathbb{B}} S(Y; Y)$, dinaturality is such that $(r \succ s) \sim (s \prec r)$ for each $r\colon Y_1 \to Y_2$ and each $s \in S(Y_1, Y_2)$.

Definition 2.16. Profunctor composition is the operation that takes two profunctors $P\colon A^{op} \times \mathbb{B} \to \mathbf{Set}$ and $Q\colon \mathbb{B}^{op} \times \mathbf{C} \to \mathbf{Set}$, into the profunctor $(P \fatsemi Q)\colon A^{op} \times \mathbf{C} \to \mathbf{Set}$ defined by their pairing quotiented by dinaturality on the middle variable,

$$(P \fatsemi Q)(X; Z) = \sum_{Y \in \mathbb{B}} P(X; Y) \times Q(Y; Z) \Big/ (\sim_Y).$$

This type of colimit is better known as a *coend*, and usually denoted by a superscripted integral sign,

$$(P \fatsemi Q)(X; Z) = \int^{Y \in \mathbb{B}} P(X; Y) \times Q(Y; Z).$$

3 EFFECTFUL STREAMS

The classical semantics for dataflow programs is given by a stream of visible outputs for every stream of inputs. In the same way, the semantics of effectful dataflow programs is given by effectful streams. The theory of monoidal streams [DLdFR22] is an analogue of the coinductive construction of streams using morphisms in a symmetric monoidal category. We adapt it to the effectful setting and employ it to give trace semantics to effectful dataflow programs.

A classical stream, $\boldsymbol{X} = (X_0, X_1, ...)$, is coinductively defined to be *(i)* an element $\boldsymbol{X}^\circ = X_0$, *(ii)* together with a stream, representing its tail, $\boldsymbol{X}^+ = (X_1, X_2, ...)$. When the stream is formed by objects of an effectful category, we can act on the first element of the stream $(M \cdot \boldsymbol{X})^\circ = M \otimes \boldsymbol{X}^\circ$ leaving the rest of the stream the same, $(M \cdot \boldsymbol{X})^+ = \boldsymbol{X}^+$. The action of a morphism $r \colon M_{\boldsymbol{g}} \to M_{\boldsymbol{f}}$ is defined by $(r \cdot \boldsymbol{f})^\circ(m, x^\circ) = \boldsymbol{f}(r(m), x^\circ)$ and $(r \cdot \boldsymbol{f})^+ = \boldsymbol{f}^+$. Effectful streams use this action to introduce a new element: an explicit memory that allows to connect each piece of the stream to the next.

Definition 3.1. An *effectful stream*, $\boldsymbol{f} \colon \boldsymbol{A} \to \boldsymbol{B}$, over an effectful category $(\mathbf{P}, \mathbf{C})$ with inputs $\boldsymbol{A} = (A_0, A_1...)$ and outputs $\boldsymbol{B} = (B_0, B_1, ...)$ is a triple consisting of

- $M_{\boldsymbol{f}} \in \mathbf{P}_{obj}$, the *memory*;
- $\boldsymbol{f}^\circ \colon \boldsymbol{A}^\circ \rightsquigarrow M_{\boldsymbol{f}} \otimes \boldsymbol{B}^\circ$, the *first action*, or *head*;
- $\boldsymbol{f}^+ \colon M_{\boldsymbol{f}} \cdot \boldsymbol{A}^+ \rightsquigarrow \boldsymbol{B}^+$, the *tail of the stream*.

Effectful streams are quotiented by dinaturality over the memory: the minimal equivalence relation $(\sim)$ such that

$$(M_{\boldsymbol{f}}, \boldsymbol{f}^\circ, \boldsymbol{f}^+) \sim (M_{\boldsymbol{g}}, \boldsymbol{g}^\circ, \boldsymbol{g}^+),$$

for each pure morphism $r \colon M_{\boldsymbol{g}} \to M_{\boldsymbol{f}}$ in $\mathbf{P}$ such that $\boldsymbol{g}^\circ \fatsemi r = \boldsymbol{f}^\circ$ and $r \cdot \boldsymbol{f}^+ \sim \boldsymbol{g}^+$. Effectful streams from $\boldsymbol{A}$ to $\boldsymbol{B}$, quotiented by dinaturality, form a set $\mathsf{Stream}(\mathbf{P}, \mathbf{C})(\boldsymbol{A}; \boldsymbol{B})$.

Remark 3.2. Definition 3.1 can be recast in coalgebraic terms. The set $\mathsf{Stream}(\mathbf{P}, \mathbf{C})(\boldsymbol{A}; \boldsymbol{B})$ of effectful streams from $\boldsymbol{A}$ to $\boldsymbol{B}$ in the effectful category $(\mathbf{P}, \mathbf{C})$ is the final fixpoint of the functor $\phi \colon [(\mathbf{C}^\omega)^{\mathrm{o}} \times \mathbf{C}^\omega, \mathbf{Set}] \to [(\mathbf{C}^\omega)^{\mathrm{o}} \times \mathbf{C}^\omega, \mathbf{Set}]$ defined by

$$\phi(\mathrm{Q})(\boldsymbol{A}, \boldsymbol{B}) \coloneqq \int^{M \in \mathbf{P}} \hom_{\mathbf{C}}(\boldsymbol{A}^\circ; M \otimes \boldsymbol{B}^\circ) \times \mathrm{Q}(M \cdot \boldsymbol{A}^+; \boldsymbol{B}^+) \,.$$

When reasoning coinductively about streams, we will sometimes need a stronger coinductive hypothesis that adds a parameter P in the first input of the stream. It is convenient to explicitly define effectful streams of this shape: *streams with parameters*.

Definition 3.3. A *stream from $\boldsymbol{X}$ to $\boldsymbol{Y}$ with a parameter $P \in \mathbf{C}_{obj}$* is a stream from $(P \cdot \boldsymbol{X}) = (P \otimes X_0, X_1, \dots)$ to $\boldsymbol{Y} = (Y_0, Y_1, \dots)$. Equivalently, it is a morphism $P \otimes X_0 \to M \otimes Y_0$ followed by a stream from $\boldsymbol{X}^+$ to $\boldsymbol{Y}^+$ with parameter M, for some object M quotiented by dinaturality.

The coinductive definitions of sequential composition and whiskering of effectful streams uses the stronger coinductive hypothesis given by streams with parameters. The definition for generic streams is, then, obtained by taking the parameter to be the monoidal unit, $P = I$.

Definition 3.4. The *sequential composition* of two streams with parameters, $\boldsymbol{f}_P \colon P \cdot \boldsymbol{X} \to \boldsymbol{Y}$ and $\boldsymbol{g}_Q \colon Q \cdot \boldsymbol{Y} \to \boldsymbol{Z}$, is the stream with parameters $(\boldsymbol{f}_P \fatsemi \boldsymbol{g}_Q) \colon (P \otimes Q) \cdot \boldsymbol{X} \to \boldsymbol{Y}$. The *whiskering* of a stream with parameters, $\boldsymbol{f} \colon P \cdot \boldsymbol{X} \to \boldsymbol{Y}$, by a stream $\boldsymbol{U}$ results on the stream $w(\boldsymbol{f}_P) \colon P \cdot \boldsymbol{U} \otimes \boldsymbol{X} \to \boldsymbol{U} \otimes \boldsymbol{Y}$. These are defined coinductively below.

$$\begin{aligned}
&(\boldsymbol{f}_P \fatsemi \boldsymbol{g}_Q)^\circ(p, q, x^\circ) = \\
&\quad \boldsymbol{f}^\circ(p, x^\circ) \rightsquigarrow m, y^\circ \\
&\quad \boldsymbol{g}^\circ(q, y^\circ) \rightsquigarrow n, z^\circ \\
&\quad \texttt{return } (m, n, z^\circ) \\
&(\boldsymbol{f} \fatsemi \boldsymbol{g})^+ = \boldsymbol{f}^+_M \fatsemi \boldsymbol{g}^+_N
\end{aligned}
\qquad
\begin{aligned}
&w(\boldsymbol{f}_P)^\circ(p, u, x^\circ) = \\
&\quad \boldsymbol{f}^\circ(p, x^\circ) \rightsquigarrow m, y^\circ \\
&\quad \texttt{return } (m, u, y^\circ) \\
&w(\boldsymbol{f}_P)^+ = w(\boldsymbol{f}_M{}^+)
\end{aligned}$$

The sequential composition of two streams, $\boldsymbol{f} \colon \boldsymbol{X} \to \boldsymbol{Y}$ and $\boldsymbol{g} \colon \boldsymbol{Y} \to \boldsymbol{Z}$, takes the parameter to be the monoidal unit, $(\boldsymbol{f} \fatsemi \boldsymbol{g}) = (\boldsymbol{f}_I \fatsemi \boldsymbol{g}_I)$. Whiskering of a stream $\boldsymbol{f} \in \mathsf{Stream}(\boldsymbol{X}; \boldsymbol{Y})$ is the stream $w(\boldsymbol{f}) \in \mathsf{Stream}(\boldsymbol{U} \otimes \boldsymbol{X}; \boldsymbol{U} \otimes \boldsymbol{Y})$ defined by whiskering with the parameter being the monoidal unit, $w(\boldsymbol{f}) = w(\boldsymbol{f}_I) \in \mathsf{Stream}(\boldsymbol{U} \otimes \boldsymbol{X}; \boldsymbol{U} \otimes \boldsymbol{Y})$.

Theorem 3.5. *Effectful streams over $(\mathbf{P}, \mathbf{C})$ form an effectful category*, $\mathsf{Stream}(\mathbf{P}, \mathbf{C})$.

Feedback is a derived operation. It takes an effectful stream $\boldsymbol{f} \colon \mathrm{wt}\boldsymbol{S} \otimes \boldsymbol{A} \to \boldsymbol{S} \otimes \boldsymbol{B}$ and returns one of type $\boldsymbol{A} \to \boldsymbol{B}$. Intuitively, this operation takes the outputs S_n of $\boldsymbol{f}$ and "feeds them back" to $\boldsymbol{f}$ as inputs in the next time step. For this operation to typecheck, $\mathrm{wt}\boldsymbol{S}$ is the operation that "delays" the stream of inputs by one time step. It is defined as $(\mathrm{wt}\boldsymbol{S})^\circ \coloneqq I$ and $(\mathrm{wt}\boldsymbol{S})^+ \coloneqq \boldsymbol{S}$.

Definition 3.6. The feedback of a stream with parameters, $f \colon (P \otimes T) \cdot \mathrm{wt}\boldsymbol{S} \otimes \boldsymbol{X} \to \boldsymbol{S} \otimes \boldsymbol{Y}$, is a stream $\mathrm{fbk}(f_{P,T}) \colon P \otimes T \cdot \boldsymbol{X} \to \boldsymbol{Y}$ defined as follows.

$$\mathrm{fbk}(f_{P,T})^\circ = f^\circ; \qquad \mathrm{fbk}(f_{P,T})^+ = \mathrm{fbk}(f^+_{M,S^\circ}).$$

Proposition 3.7. *The category of effectful streams is an effectful category with delayed feedback.*

3.1 Observational sequences

Observational sequences are sequences of morphisms $\langle t_n \mid n \in \mathbb{N} \rangle$ quotiented by *observational equivalence*: any two sequences whose finite executions coincide are equal (cf. monoidal observational sequences [SK19, DLdFR22]).

Definition 3.8. Let $(\mathbf{P}, \mathbf{C})$ be an effectful category. An *intensional sequence* between two sequences of objects, $\boldsymbol{X} = (X_0, X_1, ...)$ and $\boldsymbol{Y} = (Y_0, Y_1, ...)$, consists of a family of morphisms in $\mathbf{C}$

$$t_n \colon M_{n-1} \otimes X_n \to M_n \otimes Y_n,$$

for some sequence of objects $\boldsymbol{M} = (M_0, M_1, \dots)$. Intensional sequences form a set, $\mathsf{IntSeq}(\boldsymbol{X}, \boldsymbol{Y})$.

We define the *nth-truncation* of an intensional sequence to be the result of executing the first n morphisms *up to dinaturality*. This is to say that we consider the set of truncated sequences to be

$$\mathbf{Trunc}_n(\boldsymbol{X}, \boldsymbol{Y}) = \int^{M_0, \dots, M_n \in \mathbf{P}} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, M_i \otimes Y_i),$$

where $M_{-1} = I$ by convention, and that there exist projections $\pi_n \colon \mathsf{IntSeq}(\boldsymbol{X}, \boldsymbol{Y}) \to \mathbf{Trunc}_n(\boldsymbol{X}, \boldsymbol{Y})$.

Definition 3.9. Observational sequences are intensional sequences quotiented by observational equivalence,

$$\mathsf{ObsSeq}(\boldsymbol{X}, \boldsymbol{Y}) = \mathsf{IntSeq}(\boldsymbol{X}, \boldsymbol{Y})/(\approx).$$

Two intensional sequences, $f, g \in \mathsf{IntSeq}(\boldsymbol{X}, \boldsymbol{Y})$, are *observationally equivalent*, $f \approx g$, whenever all their truncations coincide, $\pi_n(f) = \pi_n(g)$ for each $n \in \mathbb{N}$.

The main theorem in the work of Di Lavore, de Felice and Román [DLdFR22] shows, in the monoidal case, that coinductive streams are characterized by observational sequences whenever the category satisfies a technical condition called "productivity". Intuitively, productivity ensures that, at each step, some part of the process is independent from the future and can be executed. Formally, this is ensured by every 1-stage process having a *minimal canonical* representative that gives the behaviour of the process by keeping a *maximally informative* memory.

Definition 3.10. A representative $t\colon X \to M_0 \otimes Y$ of a 1-truncated sequence, $\langle t| \in \mathbf{Trunc}_1(X,Y)$ is *minimal* if, every other representative, $\langle u| = \langle t|$, factors through it, i.e. there is a morphism $s_u\colon M_0 \to M_u$ in $\mathbf{P}$ such that $u = t \fatsemi (s_u \otimes \mathrm{id})$. Such a representative is, moreover, *canonical* if whenever

$$\langle (u \otimes \mathrm{id}) \fatsemi (a \otimes \mathrm{id})| = \langle (v \otimes \mathrm{id}) \fatsemi (b \otimes \mathrm{id})|$$

as elements of the coend $\int^{M \in \mathbf{P}} \hom(X \otimes A, M \otimes Y \otimes B)$ then

$$\langle (s_u \otimes \mathrm{id}) \fatsemi a| = \langle (s_v \otimes \mathrm{id}) \fatsemi b|$$

as elements of $\int^{M \in \mathbf{P}} \hom(M_0 \otimes A, M \otimes B)$, for $u, v \in \langle t|$, $a\colon M_u \otimes A \to M_a \otimes B$ and $b\colon M_v \otimes A \to M_b \otimes B$.

An effectful category is productive whenever every 1-stage process has a minimal canonical representative. Our interest in productivity is limited to the following characterization lemma. In the following section, we will prove that all categories we care about are indeed productive.

Definition 3.11. An effectful category $(\mathbf{P}, \mathbf{C})$ is *productive* if every 1-truncation, $\langle t| \in \mathbf{Trunc}_1(X,Y)$, has a minimal canonical representative.

Theorem 3.12. *Effectful streams over a productive effectful category* $(\mathbf{P}, \mathbf{C})$ *are isomorphic to observational sequences over the same effectful category,* $\mathsf{Stream}(\mathbf{P}, \mathbf{C}) \cong \mathsf{ObsSeq}(\mathbf{P}, \mathbf{C})$.

Productivity for copy-discard categories. We show that quasi-total conditionals and ranges give productivity. Quasi-total conditionals imply the first condition for productivity while ranges imply the second.

Proposition 3.13. *Whenever a copy-discard category,* $(\mathbf{V}, \mathbf{P})$*, has quasi-total conditionals and ranges, the effectful category of its total morphisms,* $(\mathrm{Tot}(\mathbf{P}), \mathbf{P})$*, is productive.*

Corollary 3.14. *Let* $(\mathbf{V}, \mathbf{P})$ *be a copy-discard category with quasi-total conditionals and ranges; streams coincide with observational sequences* $\mathsf{Stream}(\mathbf{P}, \mathbf{C}) \cong \mathsf{ObsSeq}(\mathbf{P}, \mathbf{C})$.

3.2 Example: the stream cipher is secure

Let us discuss security for the stream cipher protocol (Section 1.2) by giving it appropriate semantics.

For this purpose, we will use the Kleisli category of the finitely-supported distribution monad, **Stoch**. We employ two fixed sets: a finite alphabet of characters, Char, and a finite set of seeds to our random number generator, Seed. For both these sets, there exist uniform distributions, $\mathtt{unif_c}\colon () \to \mathsf{Char}$ and $\mathtt{unif_s}\colon () \to \mathsf{Seed}$.

For the set of characters, there exists moreover a nilpotent and deterministic "bitwise XOR" operation $(\oplus)\colon \mathsf{Char} \times \mathsf{Char} \to \mathsf{Char}$, for which the uniform distribution is a Sweedler integral [Swe69], meaning that XOR-ing by uniform noise results in uniform noise,

$$\left.\begin{array}{l}\mathtt{unif}() \to b \\ \mathtt{return}(x \oplus b)\end{array}\right\} = \left.\begin{array}{l}\mathtt{unif}() \to b \\ \mathtt{return}(b)\end{array}\right\}$$

Of course, it is impossible to prove that the stream cipher protocol is exactly equal to a secure channel: it can be easily seen that there exist no perfect random generators in the category of finitely-supported distributions, **Stoch**. Instead, we will prove that the protocol is "approximately equal" $(\approx)$ to the secure channel.

Assumption 3.15 (Broadbent and Karvonen, [BK23, §7.4])**.** Let $(\approx)$ be a congruence, preserved by composition and tensoring. An $(\approx)$*-pseudorandom number generator* over a finite alphabet is a deterministic morphism, $\mathtt{prng}\colon \mathsf{Seed} \to \mathsf{Char} \otimes \mathsf{Seed}$, that satisfies the following equation.

$$\left.\begin{array}{l}\mathtt{unif_s}() \to g \\ \mathtt{prng}(g) \to h, k \\ \mathtt{return}(h, k)\end{array}\right\} \approx \left.\begin{array}{l}\mathtt{unif_s}() \to g \\ \mathtt{unif_c}() \to k \\ \mathtt{return}(g, k)\end{array}\right\}$$

Proposition 3.16. *There exist no* $(=)$*-pseudorandom number generators in the category of finitary distributions,* **Stoch**.

The last ingredient we need for this interpretation is to traduce the effectful generators into modifications of a global state, this is a quite general technique [MS14]: effectful categories can be hardly modeled by a monoidal category – where interchange necessarily holds – but they can be modeled in the Kleisli category of a global state monad – where interchange can fail.

We declare global state to consist of the pair of seeds that Alice and Bob keep, Seed⊗Seed. Our semantics will consist on an effectful copy-discard functor to a category of effectful computations given by the Kleisli category of the global state promonad,

$$\mathbf{SeedStoch}(X; Y) = \mathbf{Stoch}(\mathsf{Seed} \otimes \mathsf{Seed} \otimes X; \mathsf{Seed} \otimes \mathsf{Seed} \otimes Y).$$

Definition 3.17. The *interpretation functor*,

$$[\![-]\!]\colon \mathbf{Cipher} \to (\mathbf{Set}, \mathbf{Stoch}, \mathbf{SeedStoch}),$$

is the unique effectful copy-discard functor extending the following assingment on the generators of the effectful copy-discard graph inducing **Cipher** (Example 2.10). It interprets the only object as the set of characters $[\![\mathsf{C}]\!] = \mathsf{Char}$. On values, it interprets the XOR symbol as the "bitwise XOR of characters", $[\![\oplus]\!] = (\oplus)$. It interprets the only pure morphism as the uniform distribution, $[\![\mathtt{unif}]\!] = \mathtt{unif_c}$. Finally, on effectful generators, it must provide the following interpretations,

$$\begin{array}{l}[\![\mathtt{seed}]\!]\colon \mathsf{Seed} \otimes \mathsf{Seed} \to \mathsf{Seed} \otimes \mathsf{Seed}; \\ [\![\mathtt{rand}_a]\!]\colon \mathsf{Seed} \otimes \mathsf{Seed} \to \mathsf{Seed} \otimes \mathsf{Seed} \otimes \mathsf{Char}; \\ [\![\mathtt{rand}_b]\!]\colon \mathsf{Seed} \otimes \mathsf{Seed} \to \mathsf{Seed} \otimes \mathsf{Seed} \otimes \mathsf{Char}.\end{array}$$

These are defined by the following fragments of do-notation.

$$\begin{array}{l}[\![\mathtt{rand}_a]\!](g_a, g_b) = \\ \quad \mathtt{prng}(g_a) \to h_a, k \\ \quad \mathtt{return}(h_a, g_b, k)\end{array} \qquad \begin{array}{l}[\![\mathtt{rand}_b]\!](g_a, g_b) = \\ \quad \mathtt{prng}(g_b) \to h_b, k \\ \quad \mathtt{return}(g_a, h_b, k)\end{array} \qquad \begin{array}{l}[\![\mathtt{seed}]\!](g_a, g_b) = \\ \quad \mathtt{unif}() \to g \\ \quad \mathtt{return}(g, g)\end{array}$$

Finally, we can state security for the stream cipher: it means that it is approximately equal to using secure channel that sends the

message directly from Alice to Bob and outputs random noise to an external observer.

Definition 3.18. The secure channel is the following morphism in the syntactic category **Cipher**.

$$\begin{array}{l}\texttt{secure}^{\circ}(m) = \\ \quad \texttt{unif}() \to n \\ \quad \texttt{return}(m, n); \\ \texttt{secure}^{+} = \texttt{secure}\end{array}$$

Theorem 3.19. *The interpretation of the stream cipher is approximately equal to the interpretation of the secure channel,*

$$[\![\texttt{cipher}]\!] \approx [\![\texttt{secure}]\!].$$

Proof. We will prove this statement by coinduction. Let us first compute the interpretation of the whole stream cipher from the interpretation of each one of its generators.

$$\begin{array}{l}[\![\texttt{cipher}]\!]^{\circ}(g_a, g_b, m) = \\ \quad \texttt{unif}() \to g \\ \quad \texttt{prng}(g) \to h_a, k_a \\ \quad \texttt{prng}(g) \to h_b, k_b \\ \quad \texttt{return}(h_a, h_b, \\ \quad\quad m \oplus k_a \oplus k_b, m \oplus k_a)\end{array} \qquad \begin{array}{l}[\![\texttt{cipher}]\!]^{+\circ}(g_a, g_b, m) = \\ \quad \texttt{prng}(g_a) \to h_a, k_a \\ \quad \texttt{prng}(g_b) \to h_b, k_b \\ \quad \texttt{return}(h_a, h_b, \\ \quad\quad m \oplus k_a \oplus k_b, m \oplus k_a) \\ [\![\texttt{cipher}]\!]^{++} = [\![\texttt{cipher}]\!]^{+}\end{array}$$

We can now simplify the first step of the stream cipher. We use *(i)* determinism of `unif`; *(ii)* that `unif` is a pseudorandom number generator and that XOR forms a nilpotent algebra; and *(iii)* that the uniform distribution is a Sweedler integral.

$$\left.\begin{array}{l}\texttt{unif}() \to g \\ \texttt{prng}(g_a) \to h_a, k_a \\ \texttt{prng}(g_b) \to h_b, k_b \\ \texttt{return}(h_a, h_b, \\ \quad m \oplus k_a \oplus k_b, m \oplus k_a)\end{array}\right\} \overset{(i)}{=} \left\{\begin{array}{l}\texttt{unif}() \to g \\ \texttt{prng}(g) \to h, k \\ \texttt{return}(h, h, \\ \quad m \oplus k \oplus k, m \oplus k)\end{array}\right\} \overset{(ii)}{\approx}$$

$$\left.\begin{array}{l}\texttt{unif}() \to h \\ \texttt{unif}() \to k \\ \texttt{return}(h, h, m, m \oplus k)\end{array}\right\} \overset{(iii)}{=} \left\{\begin{array}{l}\texttt{unif}() \to h \\ \texttt{unif}() \to k \\ \texttt{return}(h, h, m, k)\end{array}\right\}$$

We now rewrite the interpretation of the stream cipher as follows. By dinaturality over the `unif` statement, $[\![\texttt{cipher}]\!] \approx [\![\texttt{cipher}']\!]$.

$$\begin{array}{l}[\![\texttt{cipher'}]\!]^{\circ}(g_a, g_b, m) = \\ \quad \texttt{unif}() \to k \\ \quad \texttt{return}(g_a, g_b, m, k) \\ [\![\texttt{cipher'}]\!]^{++} = [\![\texttt{cipher}]\!]^{+}\end{array} \qquad \begin{array}{l}[\![\texttt{cipher'}]\!]^{+\circ}(g_a, g_b, m) = \\ \quad \texttt{unif}() \to g \\ \quad \texttt{prng}(g) \to h_a, k_a \\ \quad \texttt{prng}(g) \to h_b, k_b \\ \quad \texttt{return}(h_a, h_b, \\ \quad\quad m \oplus k_a \oplus k_b, m \oplus k_a)\end{array}$$

Finally, we observe that $[\![\texttt{cipher'}]\!]^{\circ} = [\![\texttt{secure}]\!]^{\circ}$; by the coinductive hypothesis, we know that

$$[\![\texttt{cipher'}]\!]^{+} = [\![\texttt{cipher}]\!] \approx [\![\texttt{secure}]\!] = [\![\texttt{secure}]\!]^{+}. \quad \square$$

4 CAUSAL PROCESSES

Causal processes are sequeqnces of morphisms $(f_n)_{n\in\mathbb{N}}$ indexed by the natural numbers, where each morphism $f_n : A_0 \otimes \cdots \otimes A_n \to B_0 \otimes \cdots \otimes B_n$ does not only represent a single step of a process, but the whole process until time n [SJ19, SK19]. This intuition leads naturally to causality condition: each morphism f_{n+1} must extend the previous one f_n; i.e. they need to have the same behaviour on the first n inputs. The notion of conditionals promotes this intuition into a formal condition: a causal process must be such that $f_n \otimes \multimap\bullet$ is a marginal for f_{n+1}.

Definition 4.1. A *causal process*, $\boldsymbol{f} : \boldsymbol{X} \to \boldsymbol{Y}$, in a copy-discard category $(\mathbf{V}, \mathbf{P})$, is a family of morphisms

$$f_n : X_0 \otimes \ldots \otimes X_n \to Y_0 \otimes \ldots \otimes Y_n$$

that are *causal*: each $f_n \otimes \multimap\bullet$ is a marginal of f_{n+1}. Explicitly, there must exist morphisms $c_n : Y_0 \otimes \ldots \otimes Y_n \otimes X_0 \otimes \ldots \otimes X_{n+1} \to Y_{n+1}$, the *conditionals*, that satisfy the following equation[1].

$$f_{n+1}(\vec{x}_{n+1}) \;\to\; \vec{y}_{n+1}\} = \left\{\begin{array}{l} f_n(\vec{x}_n) \;\to\; \vec{y}_n \\ c_n(\vec{y}_n, \vec{x}_{n+1}) \;\to\; y_{n+1}\end{array}\right\}$$

String diagrams for this condition are in Figure 1.

Note that the conditionals c_n of the morphisms f_n don't need to be quasi-total. This ensures that f_{n+1} is allowed to fail on an input $(x_0, \ldots, x_n, x_{n+1})$, even if f_n did not fail on $(x_0, \ldots, x_n)$.

This section characterises effectful streams for copy-discard categories with quasi-total conditionals and ranges as causal processes. Examples include the categories **Set** of sets and functions, **Rel** of relations, **Par** of partial functions, **Stoch** of finitary probabilistic processes and **ParStoch** of finitary partial probabilistic processes, which we study in detail.

While the coinductive definition of effectful streams is convenient for defining and implementing them, the characterisation as causal processes allows one to avoid working up to dinaturality and easily compute traces (Section 5.3). Causal processes form a symmetric monoidal category where compositions, identities and monoidal products are defined component-wise by those in the base category.

Proposition 4.2. *Causal processes over a copy-discard category* $(\mathbf{V}, \mathbf{P})$ *with quasi-total conditionals form a copy-discard category,* $\mathrm{Causal}(\mathbf{V}, \mathbf{P})$*, whose objects are streams* $\boldsymbol{A} = (A_0, A_1, \ldots)$ *of objects of* $\mathbf{P}$ *and whose morphisms* $\boldsymbol{f} : \boldsymbol{A} \to \boldsymbol{B}$ *are causal processes with inputs in* $\boldsymbol{A}$ *and outputs in* $\boldsymbol{B}$*.*

Proof. See Appendix E for the details. □

Each causal process $\boldsymbol{f} : \boldsymbol{X} \to \boldsymbol{Y}$ has a sequence $c_n : X_0 \otimes \cdots \otimes X_n \otimes Y_0 \otimes \cdots \otimes Y_{n-1} \to Y_n$ of conditionals of f_n. Each of these sequences defines a stream that, at every step n, has in memory all the inputs and outputs until time $n-1$, receives the input at time n and produces the output at time n using c_n. The resulting stream is independent of the choice of conditionals and this mapping defines a monoidal functor. The proof works with observational sequences, which are isomorphic to streams whenever there are conditionals and ranges (Theorem 3.12).

Proposition 4.3. *For a copy-discard category* $(\mathbf{V}, \mathbf{P})$ *with quasi-total conditionals and ranges, there is an identity-on-objects monoidal functor* $\mathrm{obs} : \mathrm{Causal}(\mathbf{P}) \to \mathrm{ObsSeq}(\mathrm{Tot}(\mathbf{P}), \mathbf{P})$*.*

Proof. See Appendix E. □

[1] For convenience, we denote with $\vec{x}_n$ a list $(x_0, \ldots, x_n)$ of elements of $X_0 \otimes \cdots \otimes X_n$.

Proposition 4.4. *An observational sequence $\langle t_n \mid n \in \mathbb{N}\rangle : \boldsymbol{X} \to Y$ defines a causal process by executing the components t_n in sequence.*

$$\begin{aligned} &f_n(x_0, \dots, x_n) = \\ &\quad t_0(x_0) \to (m_0, y_0) \\ &\quad \dots \\ &\quad t_n(m_{n-1}, x_n) \to (m_n, y_n) \\ &\quad \texttt{return}(y_0, \dots, y_n) \end{aligned}$$

This assingment extends to a monoidal functor

$$\mathrm{proc} \colon \mathrm{ObsSeq}(\mathrm{Tot}(\mathbf{P}), \mathbf{P}) \to \mathrm{Causal}(\mathbf{P}).$$

Proof. Lemma E.4 shows that proc is well-defined. Together with fully-faithfulness of obs (Proposition 4.5), we obtain that proc is a monoidal functor. □

This sequence is independent of the particular representatives t_n chosen for the observational sequence (Lemma E.3). Proposition 4.5 show that this mapping makes the functor obs of Proposition 4.3 fully-faithful, i.e. bijective on morphisms.

Proposition 4.5. *For a copy-discard category $(\mathbf{V}, \mathbf{P})$ with quasi-total conditionals and ranges, the functor obs is fully-faithful.*

Proof. See Lemmas E.7 and E.9. □

Identity-on-objects fully-faithful functors are isomorphisms and we obtain that streams is isomorphic to causal processes by applying Theorem 3.12.

Theorem 4.6. *For a copy-discard category $(\mathbf{V}, \mathbf{P})$ with quasi-total conditionals and ranges, the category of effectful streams is monoidal and isomorphic to the monoidal category of causal processes,*

$$\mathrm{Stream}(\mathrm{Tot}(\mathbf{P}), \mathbf{P}) \cong \mathrm{Causal}(\mathbf{P}).$$

As a consequence of this result, we recover the characterisations of cartesian and stochastic streams as causal processes [DLdFR22]. All the examples in this section are Kleisli categories over **Set**, so they inherit the copy-discard structure ($\multimap$, $\to$) from it. In these cases, we denote the monoidal product as $\times$.

Example 4.7 ([DLdFR22, Section 6]). Streams over a cartesian category coincide with causal stream functions [SJ19] because, in the cartesian case, conditionals always exist and the causality condition simplifies. Since the outputs are independent, it is sufficient to specify, for each n, the n-th output depending on the first n inputs. This coincides with the classical notion of causality for stream functions [Ran58]. The details are in Lemma E.10.

A *cartesian causal process* $f \colon \boldsymbol{A} \to \boldsymbol{B}$ is a family of morphisms $f_n \colon A_0 \times \cdots \times A_n \to B_n$ indexed by the natural numbers, $n \in \mathbb{N}$. These were described as the coKleisli category of the non-empty list comonad [UV08], but this characterisation works only when the base category $\mathbf{C}$ is cartesian [DLdFR22, Theorem 6.1].

Example 4.8 ([DLdFR22, Section 7]). Streams over the Kleisli category of the finitary distribution monad, **Stoch**, yield controlled stochastic processes [FR75, Ros96]. The monoidal category **Stoch** has conditionals [Fri20] that are quasi-total because all morphisms are total. Explicitly, a conditional of a morphism $f \colon X \to A \times B$ in **Stoch** is given by

$$c(b \mid x, a) = \frac{f(a, b \mid x)}{\sum_{b' \in B} f(a, b' \mid x)}$$

whenever defined for the given x and a, and by any arbitrary distribution over b otherwise. The monoidal category **Stoch** also has ranges [DLdFR23, Proposition 9.9]. For a morphism $m \colon X \to A$ its range r is the deterministic morphism defined as

$$r(x, a) = \begin{cases} (x, a), & \text{if } m(a \mid x) > 0; \\ (x, a_x), & \text{if } m(a \mid x) = 0; \end{cases}$$

for some elements $a_x \in A$ such that $m(a_x \mid x) > 0$. Note that r is also total because every morphism is and factors as $\mathrm{id} \fatsemi r$.

A probabilistic causal process $f \colon \boldsymbol{A} \to \boldsymbol{B}$ is a family of functions $f_n \colon A_0 \times \cdots \times A_n \to \mathrm{D}(B_0 \times \cdots \times B_n)$ indexed by the natural numbers such that $f_{n+1} \fatsemi (\mathrm{id} \times \multimap_{n+1}) = f_n \times \multimap_{n+1}$. The causality condition is simplified because all morphisms are total.

The distinction between morphisms that are pure and effectful is crucial to the characterisation of Theorem 4.6. It allows the result to hold even when the morphisms in the base category are not necessarily total. In fact, the more naive construction of monoidal streams over the category **Rel** of relations collapses: every two parallel morphisms are equal because, thanks to the compact closed structure, a stream can always wait and produce nothing.

We study in detail the cases of relational, partial and partial stochastic causal processes. We refer the reader to Appendix E.1 for the proofs of the results about conditionals and ranges in these categories.

4.1 Relational causal processes

Consider the copy-discard category $(\mathbf{Set}, \mathbf{Rel})$ where objects are sets and morphisms $X \to Y$ are relations between X and Y, i.e. functions to the powerset, $X \to \wp(Y)$. This category has conditionals (see Figure 4) and they are quasi-total – in fact, every morphism in **Rel** is quasi-total.

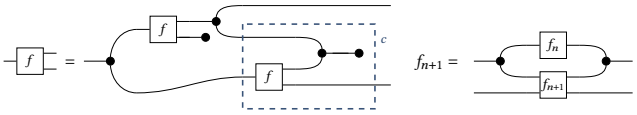


Figure 4: A quasi-total conditional c of f in Rel, where $\multimap$ denotes the opposite relation to $\to$ and the causality condition for relational processes.

Proposition 4.9. *A relational causal process $\boldsymbol{f} \colon \boldsymbol{A} \to \boldsymbol{B}$ is a family of functions $f_n \colon A_0 \times \cdots \times A_n \to \wp(B_0 \times \cdots \times B_n)$ indexed by the natural numbers satisfying the equation in Figure 4.*

Relations also have ranges that are the identity on all the possible input-output pairs and empty otherwise. For a relation $m \colon X \to A$, its range is given by the partial function $r' \colon X \times A \to R$ and the function $\iota \colon R \to X \times A$, with $R \coloneqq \{(x, a) \in X \times A : a \in m(x)\}$, defined as

$$r'(x, a) = \begin{cases} \{(x, a)\}, & \text{if } a \in m(x); \\ \emptyset, & \text{otherwise;} \end{cases} \quad \text{and} \quad \iota(x, a) = \{(x, a)\}.$$

Lemma 4.10. *The monoidal category of relations* **Rel** *has quasi-total conditionals (Figure 4) and ranges.*

Proof. See Appendix, Lemmas E.11 and E.12. □

As a consequence of Lemma 4.10 and Proposition 4.2 relational causal processes form a monoidal category, and, applying Lemma 4.10 and Theorem 4.6, this category is isomorphic to the monoidal category of streams over the effectful category of relations and total relations, (**tRel**, **Rel**).

Corollary 4.11. *Relational causal processes form a monoidal category* Causal(**Rel**) *that is isomorphic to* Stream(**tRel**, **Rel**).

4.2 Partial deterministic causal processes

We now consider the category of partial functions, **Par**, whose objects are sets and whose morphisms $X \to Y$ are partial functions, i.e. functions of the form $X \to Y + 1$. Alternatively, **Par** is the Kleisli category of the *Maybe* monad over **Set**.

Lemma 4.12. *The monoidal category of partial functions* **Par** *has quasi-total conditionals (Figure 5) and ranges.*

Proof. Lemma E.13 checks that the conditionals defined in Figure 5 are quasi-total and satisfy the property of conditionals. Lemma E.14 checks the properties of ranges. □

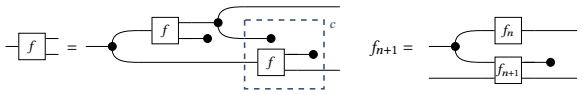


Figure 5: Quasi-total conditionals and causality for deterministic partial processes.

A partial causal process $\boldsymbol{f} \colon \boldsymbol{A} \to \boldsymbol{B}$ is a family of functions $f_n \colon A_0 \times \cdots \times A_n \to (B_0 \times \cdots \times B_n) + 1$ indexed by the natural numbers satisfying the equation in Figure 5.

As a consequence of Lemma 4.12 and Proposition 4.2 partial causal processes form a monoidal category; applying Lemma 4.12 and Theorem 4.6, this category is isomorphic to the monoidal category of streams over the effectful category of partial functions and functions, (**Set**, **Par**).

Corollary 4.13. *Partial deterministic causal processes form a monoidal category* Causal(**Par**) *that is isomorphic to* Stream(**Set**, **Par**).

4.3 Partial stochastic streams

Consider the Kleisli category **ParStoch** of the finitary subdistribution monad on **Set**. Its objects are sets and morphisms $X \to Y$ are functions $f \colon X \to \mathrm{D}_{\leq 1}(Y)$ that assign to each x and y a number $f(y \mid x) \in [0, 1]$, the probability of y given x, such that *(i)* the total probability mass is at most 1, $\sum_{y \in Y} f(y \mid x) \leq 1$ and *(ii)* the support $\{y \in Y : f(y \mid x) > 0\}$ is finite.

Lemma 4.14. *The monoidal category* **ParStoch** *has quasi-total conditionals and ranges (see Appendix, Lemma E.15.).*

The causality condition in **ParStoch** remains the general one of Figure 1. As a consequence of [DLR23, Proposition 2.13] and Proposition 4.2, partial stochastic causal processes form a monoidal category and, applying Lemma E.16 and Theorem 4.6, we obtain that it is isomorphic to streams over (**Stoch**, **ParStoch**).

Corollary 4.15. *Partial stochastic causal processes form a monoidal category,* Causal(**ParStoch**)*, that is isomorphic to the category of streams over partial stochastic functions* Stream(**Stoch**, **ParStoch**).

5 EFFECTFUL MEALY MACHINES

This section employs effectful streams as semantic universe for state machines with an initial state, also known as *Mealy machines*. We introduce Mealy machines in an effectful copy-discard category, and show that they compose forming an effectful copy-discard category themselves (Proposition 5.3). The categorical structure naturally leads to definitions of trace equivalence (Definition 5.5) and bisimulation (Definition 5.8) for effectful Mealy machines.

Section 5.3 concludes by checking our definitions against the coalgebraic literature: when the effectful copy-discard category is (**Set**, Tot(kl(T)), kl(T)) for a commutative monad T on **Set** preserving weak pullbacks, effectful bisimulation coincides with the classical notion of coalgebraic bisimulation (Proposition 5.17). For the monads that define deterministic, partial, non-deterministic and stochastic Mealy machines, we show that effectful trace equivalence coincides with existing notions of trace semantics.

Our definition of effectful Mealy machine originates from the work of Katis, Sabadini and Walters [KSW97] where the authors define monoidal Mealy machines and their bicategory.

Definition 5.1. An *effectful Mealy machine* in an effectful copy-discard category, $(\mathbf{V}, \mathbf{P}, \mathbf{C})$, taking inputs on $A \in \mathrm{Obj}(\mathbf{C})$ and producing outputs in $B \in \mathrm{Obj}(\mathbf{C})$, is a triple (U, i, f) consisting of a *state space* $U \in \mathrm{Obj}(\mathbf{C})$, an initial state $i \colon I \to U$ in $\mathbf{C}$, and a *transition morphism*, $f : U \otimes A \rightsquigarrow U \otimes B$ in $\mathbf{C}$.

Associativity of composition of effectful Mealy machines only holds up to isomorphism, there are two options: consider a *bicategory* of them [KSW97]; or consider Mealy machines modulo the equivalence relation $\cong$ given by isomorphisms of Mealy machines [KSW99], defined below. To simplify presentation, we choose the second possibility.

A homomorphism of effectful Mealy machines is a value morphism between their state spaces that preserves the transition structure and the initial state (Figure 10).

Definition 5.2. A homomorphism of effectful Mealy machines with the same inputs and outputs, $(U, i, f) \Rightarrow (V, j, g)$, is a value morphism $\alpha \colon U \to V$ such that $i \fatsemi \alpha = j$ and moreover

$$\left.\begin{array}{l} f(i, x) \rightsquigarrow u_1, y \\ \texttt{return}(\alpha(u_1), y) \end{array}\right\} = \left\{\begin{array}{l} g(\alpha(i), x) \rightsquigarrow v_1, y \\ \texttt{return}(v_1, y) \end{array}\right\}.$$

Effectful Mealy machines quotiented by isomorphism assemble into a category via the *state construction* [KSW02, DLGR+23]. We show that this construction also preserve the effectful structure.

Proposition 5.3. *Effectful Mealy machines over an effectful copy-discard category* $(\mathbf{V}, \mathbf{P}, \mathbf{C})$ *and quotiented by isomorphism of the state space form an effectful copy-discard category,* **Mealy**$(\mathbf{V}, \mathbf{P}, \mathbf{C})$*, with objects the objects of* $\mathbf{C}$ *and morphisms* $(U, i, f) \colon A \to B$ *are effectful Mealy machines. The composition of two Mealy machines tensors both state spaces and both initial states (Figure 11). Sequential composition* $\mathsf{seq}(f, g)$ *and whiskering* $\mathsf{whisk}_C(f)$ *are specified below.*

$$\begin{array}{l} \mathsf{seq}(f,g)(\mathtt{m}_0,\mathtt{n}_0,\mathtt{a}) = \\ \quad f(\mathtt{m}_0,\mathtt{a}) \rightsquigarrow \mathtt{m}_1,\mathtt{b} \\ \quad g(\mathtt{n}_0,\mathtt{b}) \rightsquigarrow \mathtt{n}_1,\mathtt{c} \\ \quad \mathtt{return}\ (\mathtt{m}_1,\mathtt{n}_1,\mathtt{c}) \end{array} \qquad \begin{array}{l} \mathsf{whisk}_C(f)(\mathtt{m}_0,\mathtt{a},\mathtt{c}) = \\ \quad f(\mathtt{m}_0,\mathtt{c}) \rightsquigarrow \mathtt{m}_1,\mathtt{b} \\ \quad \mathtt{return}\ (\mathtt{m}_1,\mathtt{b},\mathtt{c}). \end{array}$$

Example 5.4. We recast the components of the stream cipher (Section 1.2) as effectful Mealy machines. There is an agent, $\mathtt{act} \colon 1 \times$

$C \to 1 \times C$, that receives a message and encrypts or decrypts it using the operation, XOR. Another agent, $\texttt{attack}\colon C^* \times C \to C^* \times C^*$, listens to the broadcast messages and records them in memory.

```
act(m) =                attack(m,x,y) =
  rand() ⇝ k              return((x :: m), (x :: m))
  return(m ⊕ k)
```

These agents are given a initial states $s_0\colon 1 \to 1$ and $c_0\colon 1 \to C^*$ defined by $s_0 = \texttt{seed}$ and $c_0 = \texttt{return([])}$. With these transition morphisms and initial states, we can define the following Mealy machines that we will use to recover the cipher protocol.

$\texttt{alice} \coloneqq (1, s_0, \texttt{act}) \quad \texttt{bob} \coloneqq (1, \mathrm{id}, \texttt{act}) \quad \texttt{eve} \coloneqq (C^*, c_0, \texttt{attack})$

5.1 Compositional effectful traces

Every effectful Mealy machine induces an effectful stream that represents its execution. An object A can be repeated to form a stream $[\![A]\!]$, defined by $[\![A]\!]^\circ = A$ and $[\![A]\!]^+ = [\![A]\!]$. Analogously, a transition morphism, $f\colon U \otimes A \to U \otimes B$ with state space U, can be repeated to form an effectful stream $[\![f]\!]\colon U \cdot [\![A]\!] \to [\![B]\!]$ defined by $[\![f]\!]^\circ = f$ and $[\![f]\!]^+ = [\![f]\!]$. When the transition morphism has an initial state, s_0, it defines a Mealy machine, and we can use it to construct an effectful stream that represents the execution trace of the Mealy machine. The operation $\cdot$ attaches the initial state i at the beginning of the execution of the machine f and gives its trace.

Definition 5.5. The *trace* $\mathrm{tr}(U, i, f)\colon [\![A]\!] \to [\![B]\!]$ of an effectful Mealy machine $(U, i, f)\colon A \to B$ is the effectful stream defined by $\mathrm{tr}(U, i, f) = i \cdot [\![f]\!]$. We say that two Mealy machines are *trace-equivalent* if their traces coincide.

The trace is compositional: it respects the categorical structure of effectful Mealy machines.

Proposition 5.6. *The trace of effectful Mealy machines defines an effectful functor* $\mathrm{tr}\colon \mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C}) \to \mathrm{Stream}(\mathbf{P}, \mathbf{C})$.

Example 5.7. The effectful Mealy machines defined in Example 5.4 can be composed into the cipher protocol, preserving their semantics. The cipher protocol takes a message as input and encrypts it with alice. The encrypted message is then broadcast and sent to the receiver bob, who decrypts it. The protocol allows for a listener, eve, who receives the encrypted messages and stores them in memory.

```
cipher(m) =
  alice(m) ⇝ b
  bob(b) ⇝ m'
  eve(b) ⇝ c
  return (m',c)
```

5.2 Bisimulation implies trace equivalence

Morphisms of effectful Mealy machines give rise to an equivalence relation that is different from the one of trace. More precisely, it is weaker than trace equivalence. In the case of Kleisli categories of weak pullback-preserving commutative monads on **Set**, this equivalence relation coincides with the classical notion of coalgebraic bisimulation. Intuitively, two Mealy machines should be bisimilar when they map to the same bisimulation quotient. This condition translates into the existence of a cospan of homomorphisms between them. Bisimilarity is, usually, an equivalence relation and, in particular, it is transitive. However, cospans do not compose in general, but they do when the category has pullbacks. In general, we keep sequences of cospans to specify bisimilarity.

Definition 5.8. Two effectful Mealy machines are *bisimilar* if they are connected by homomorphisms. Explicitly, for two machines $(U, i, f), (V, j, g)\colon A \to B$, there need to be sequences (U_k, i_k, f_k) and (V_k, j_k, g_k), for $k = 0, \ldots, n$, with $(U_0, i_0, f_0) = (U, i, f)$ and $(V_n, j_n, g_n) = (V, j, g)$, together with morphisms $\alpha_k\colon (U_k, i_k, f_k) \Rightarrow (V_k, j_k, g_k)$ and $\beta_k\colon (U_{k+1}, i_{k+1}, f_{k+1}) \Rightarrow (V_k, j_k, g_k)$.

Homomorphisms of effectful Mealy machines compose sequentially and in parallel, making $\mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C})$ a 2-dimensional category. We show that they also preserve the effectful structure, making $\mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C})$ an effectful 2-category. This is an effectful category where the hom-sets carry the structure of a category; this gives a strict counterpart to *effectful bicategories* [PS23].

Proposition 5.9. *Let* $(\mathbf{V}, \mathbf{P}, \mathbf{C})$ *be a strict effectful copy-discard category (or consider its strictification). Homomorphisms of effectful Mealy machines make* $\mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C})$ *an effectful 2-category.*

With this result, we can restate bisimulation: two effectful Mealy machines of type $A \to B$ are bisimilar if they belong to the same connected component in the hom-category $\mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C})(A, B)$.

Theorem 5.10. *If there is a morphism between two effectful Mealy machines, then they are trace equivalent. In particular, bisimulation implies effectful trace equivalence.*

The effectful category $\mathrm{Stream}(\mathbf{P}, \mathbf{C})$ can be trivially made into an effectful 2-category by only adding identity 2-cells. Theorem 5.10 ensures that the trace extends to an effectful 2-functor.

Corollary 5.11. *The trace effectful functor extends to an effectful 2-functor* $\mathrm{tr}\colon \mathbf{Mealy}(\mathbf{V}, \mathbf{P}, \mathbf{C}) \to \mathrm{Stream}(\mathbf{P}, \mathbf{C})$.

5.3 Case Study: T-Mealy Machines

In this section, we check our semantics framework against state-machines. We consider partial, non-deterministic and probabilistic transducers, also known as *Mealy machines*. In order to deal with all of them at once, we found convenient to model effects as monads $\mathsf{T}\colon \mathbf{Set} \to \mathbf{Set}$. For the sake of simplicity, we assume T to be *commumative*: this guarantees that the associated Kleisli category $\mathsf{kl}(\mathsf{T})$ is monoidal and thus to apply the results in Section 4. Hereafter, we will always work in the effectful copy-discard category $(\mathbf{Set}, \mathrm{Tot}(\mathsf{kl}(\mathsf{T})), \mathsf{kl}(\mathsf{T}))$.

Remark 5.12. Modulo the initial state i, Mealy machines are exactly coalgebras [Rut00] for the functor $\mathsf{T}(Id \otimes B)^A$. Similarly, the morphisms are coalgebra homomorphisms.

T-traces. We can instantiate the construction of effectful trace in this case. It is worth to mention that the effectful streams arising as traces of Mealy machines are of a rather simple shape. Indeed, for all sets A, $[\![A]\!]$ is nothing else than A^ω, namely the sets of all infinite sequences $(a_0, a_1, \ldots)$ with each $a_i \in A$. For later use, we fix A^n to be the set of n-uple $(a_0, a_1, \ldots, a_{n-1})$. For a Mealy machine (U, i, f), the effectful stream $\mathrm{tr}(U, i, f)\colon A^\omega \to B^\omega$ is the observational sequence (Definition 3.9) having the family of objects $U^\omega = (U, U, \ldots)$ as memories, intuitively meaning that the

memory object is always the same state space, and the family of $\mathsf{kl}(\mathsf{T})$-morphisms $\langle t_n \mid n \in \mathbb{N} \rangle$ defined as

$$t_0 = (i \otimes id_A) \fatsemi f : I \otimes A \to U \otimes B \qquad t_{n+1} = f : U \otimes A \to U \otimes B \,.$$

When $\mathsf{kl}(\mathsf{T})$ has quasi-total conditionals and ranges, thanks to Theorem 4.6, $\mathsf{tr}(U, i, f)$ can be characterised as a causal process. The family of morphisms defined by Proposition 4.4 reduces for $\mathsf{tr}(U, i, f)$ to the following family of $\mathsf{kl}(\mathsf{T})$-morphisms $f_n : A^{n+1} \to B^{n+1}$.

$$f_n \coloneqq \left.\begin{array}{l} t_0(a_0) \to (x_1, b_0) \\ f(x_1, a_1) \to (x_2, b_1) \\ \cdots \\ f(x_n, a_n) \to (x_{n+1}, b_n) \\ \texttt{return } (\overrightarrow{b_n}) \end{array}\right\} \tag{1}$$

Hereafter, we illustrate that by instantiating T to some monads, one recovers existing notions of Mealy machines and their semantics. To spell out their traces, one can safely exploit (1) since, for all the monads we consider, $\mathsf{kl}(\mathsf{T})$ has quasi-total conditionals and ranges, as shown in Section 4.

Example 5.13 (Mealy machines). We start by taking T to be the identity monad $\mathsf{Id} : \mathbf{Set} \to \mathbf{Set}$. In this case, $\mathsf{kl}(\mathsf{T})$ is $\mathbf{Set}$ and a T-machine $(U, i, f) : A \to B$ consists of an initial state $i(\bullet) \in U$, hereafter referred as u_0, and a transition function $f : U \times A \to U \times B$. The causal process $\mathsf{tr}(U, i, f) : A^\omega \to B^\omega$ is the family of functions $f_n : A^{n+1} \to B^{n+1}$ mapping $(a_0, \ldots, a_n)$ into $(b_0, \ldots, b_n)$ where $(u_{k+1}, b_k) = f(u_k, a_k)$ for all $k \leq n$. This is exactly the standard semantics by means of causal functions given, e.g., in [SJ19].

Example 5.14 (Partial Mealy machines). For T the maybe monad, $\mathsf{Id} + 1 : \mathbf{Set} \to \mathbf{Set}$, $\mathsf{kl}(\mathsf{T})$ is the category $\mathbf{Par}$ of partial functions. In a T-machine $(U, i, f) : A \to B$, both $i : I \to U$ and $f : U \times A \to U \times B$ are partial functions. The partial causal process (see Section 4.2) $\mathsf{tr}(U, i, f) : A^\omega \to B^\omega$ is the family of partial functions $f_n : A^{n+1} \to B^{n+1}$ mapping $(a_0, \ldots, a_n)$ into $(b_0, \ldots, b_n)$ whenever $i(\bullet) = u_0$ and, for all $k \leq n$, $(u_{k+1}, b_k) = f(u_k, a_k)$.

Example 5.15 (Non-deterministic Mealy machines). For T the powerset monad $\wp : \mathbf{Set} \to \mathbf{Set}$, $\mathsf{kl}(\mathsf{T})$ is the category $\mathbf{Rel}$ and T-machines $(U, i, f) : A \to B$ are exactly non-deterministic Mealy machines as defined in [BK08]. These are a set of initial states $i(\bullet) \subseteq U$ and a transition relation $f : U \times A \to U \times B$. The relational causal process (see Section 4.1) $\mathsf{tr}(U, i, f) : A^\omega \to B^\omega$ is the family of relations $f_n : A^{n+1} \to B^{n+1}$ relating $(a_0, \ldots, a_n) \in A^{n+1}$ to $(b_0, \ldots, b_n)$ whenever $\exists u_0, \ldots u_{n+1} \,.\, u_0 \in i(\bullet) \wedge \forall k \leq n \,.\, (u_{k+1}, b_k) \in f(u_k, a_k)$.

Example 5.16 (Partial stochastic Mealy machines). Finally, we take T to be the finitary subdistribution monad $\mathsf{D}_{\leq 1} : \mathbf{Set} \to \mathbf{Set}$. In a T-machines $(U, i, f) : A \to B$, $i(\bullet)$ is a subdistribution in $\mathsf{D}_{\leq 1}(U)$ and for all $(u, a) \in U \times A$, $p(u, a)$ is a subdistribution in $\mathsf{D}_{\leq 1}(U \times B)$. The partial stochastic causal process (see Section 4.3) $\mathsf{tr}(U, i, f) : A^\omega \to B^\omega$ is the family of $\mathsf{kl}(\mathsf{D})$-morphisms $f_n : A^{n+1} \to B^{n+1}$ mapping $(a_0, \ldots, a_n) \in A^{n+1}$ into the distribution in $\mathsf{D}(B^{n+1})$ assigning to each $(b_0, \ldots, b_n)$ the probability

$$\sum_{(u_0, \ldots, u_{n+1})} i(u_0 \mid \bullet) \cdot \prod_{k \leq n} p(\,(u_{k+1}, b_k) \mid (u_k, a_k)\,) \,.$$

T-*bisimilarity.* The notion of morphism of T-Mealy machines gives rise to an equivalence relation different from the one of traces.

When instantiating T to the monads in the previous examples, our definition of bisimilarity captures the expected notions of bisimilarities. In fact, $\equiv$ coincides with coalgebraic bisimilarity [Rut00] under reasonable assumptions. The reader should be aware that in the literature of coalgebras there are several notions of bisimilarity (see [Sta09] for a complete overview) but they all coincide when restricting to *weak-pullback preserving* endofunctors on $\mathbf{Set}$.

As discussed in Remark 5.12, T-Mealy machines are coalgebras for the functor $\mathsf{T}(Id \otimes B)^A$ modulo the presence of initial states $i : I \to \mathsf{T}(U)$. Since coalgebraic bisimilarity is defined on states, i.e., elements of U and not of $T(U)$, to formally state the correspondence we assume that i factors as $i' : I \to X$ and $\eta_X : X \to \mathsf{T}(X)$, intuitively meaning that i is actually an element of U. We do not loose generality with this assumption as the initial state can always be added to the transition morphism by adding an extra state, $\bullet$. The initial state of this transformed machine will be the state $\bullet$.

Proposition 5.17. *Let* $\mathsf{T} : \mathbf{Set} \to \mathbf{Set}$ *be a commutative monad preserving weak pullbacks and let* $(U, i, f), (V, j, g) : A \to B$ *be two* T*-Mealy machines such that* i *and* j *factors respectively as* $i' \fatsemi \eta_X$ *and* $j' \fatsemi \eta_Y$*. Then* $i'(\bullet)$ *and* $j'(\bullet)$ *are bisimilar in the sense of [Rut00] if and only if* $(U, i, f) \equiv (V, j, g)$*.*

It is well known that in non-deterministic systems bisimilarity entails trace equivalence. The same happens for coalgebras (when traces are defined as in both [HJS07] and [SBBR10]) and for our T-Mealy machines, as a consequence of Theorem 5.10.

6 CONCLUSIONS

We have introduced effectful streams, a notion of trace for effectful programs and effectful state machines. Effectful streams recast the classical coinductive description of streams, but they use the morphisms of an effectful copy-discard category. Effectful copy-discard categories are a refined version of the categorical semantics for a programming language with a value/computation distinction. These programs can be understood as a formal syntax for the morphisms of effectful copy-discard categories and effectful streams become the execution trace of the programs.

Do-notation for effectful streams is not only a convenient syntax for the description of effectful streams; together with dinaturality and coinduction, it allows us to reason about the behaviour of potentially infinite iterative processes. Our running example, the stream cipher, is an illustration: reasoning about the safety of the one-time pad extends coinductively to a proof of safety for the whole stream cipher. Many effectful programs (like servers, or operating systems) are not designed to stop but to run forever; further work could try to extend our techniques to these cases.

We have shown that effectful streams are can be characterized by causal processes, a generalized notion of causal function sequence that can be defined in any monoidal category with conditionals. Conditionals come from categorical probability, where they are used to alter the causal structure of a probabilistic process by Bayesian inversion; the fact that the same structure appears in the study of causal functions makes us think that conditionals may play a role in many other aspects of categorical semantics apart from their

probabilistic interpretation. Further work may explore applications of Markov categories to traces and causality.

We have seen how effectful streams allow us to introduce notions of bisimulation and trace equivalence that particularize to the well-known monadic cases while generalizing them to effectful categories. The bicategorical structure of effectful Mealy machines seems to be of particular interest.

REFERENCES

[BBLM17] Giovanni Bacci, Giorgio Bacci, Kim G. Larsen, and Radu Mardare. On the metric-based approximate minimization of markov chains. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 104:1–104:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.ICALP.2017.104, doi:10.4230/LIPICS.ICALP.2017.104.

[BDEP97] Richard Blute, Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled Markov processes. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 149–158, 1997. doi:10.1109/LICS.1997.614943.

[BHP+19] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Pawel Sobocinski, 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.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BK23] Anne Broadbent and Martti Karvonen. Categorical composable cryptography: extended version. *Log. Methods Comput. Sci.*, 19(4), 2023. URL: https://doi.org/10.46298/lmcs-19(4:30)2023, doi:10.46298/LMCS-19(4:30)2023.

[BLLL23] G. Boccali, A. Laretto, F. Loregian, and S. Luneia. Completeness for categories of generalized automata. In Paolo Baldan and Valeria de Paiva, editors, *10th Conference on Algebra and Coalgebra in Computer Science (CALCO 2023)*, volume 270 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CALCO.2023.20.

[BSS21] Filippo Bonchi, Alexandra Silva, and Ana Sokolova. Distribution Bisimilarity via the Power of Convex Algebras. *Logical Methods in Computer Science*, 17(3), 07 2021. doi:10.46298/lmcs-17(3:10)2021.

[BSV22] Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The Theory of Traces for Systems with Nondeterminism, Probability, and Termination. *Logical Methods in Computer Science*, 18(2), 06 2022. doi:10.46298/lmcs-18(2:21)2022.

[CDVP21] Titouan Carette, Marc De Visme, and Simon Perdrix. Graphical language with delayed trace: Picturing quantum computing with finite memory. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021.

[CG99] Andrea Corradini and Fabio Gadducci. An Algebraic Presentation of Term Graphs, via GS-Monoidal Categories. *Applied Categorical Structures*, 7(4):299–331, 1999. doi:10.1023/A:1008647417502.

[CKWW08] Aurelio Carboni, Max Kelly, Robert FC Walters, and Richard J Wood. Cartesian bicategories II. *Theory and Applications of Categories*, 19(6):93–124, 2008.

[CLZ23] Titouan Carette, Louis Lemonnier, and Vladimir Zamdzhiev. Central submonads and notions of computation: Soundness, completeness and internal languages. In *LICS*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175687.

[CP20] Thomas Colcombet and Daniela Petrişan. Automata minimization: a functorial approach. *Logical Methods in Computer Science*, 16, 2020.

[DDLP06] Vincent Danos, Josée Desharnais, François Laviolette, and Prakash Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204(4):503–523, 2006.

[DLdFR22] Elena Di Lavore, Giovanni de Felice, and Mario Román. Monoidal streams for dataflow programming. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533365.

[DLdFR23] Elena Di Lavore, Giovanni de Felice, and Mario Román. Coinductive streams in monoidal categories, 2023. arXiv:2212.14494.

[DLGL17] Ugo Dal Lago, Francesco Gavazzo, and Paul Blain Levy. Effectful applicative bisimilarity: Monads, relators, and Howe's method. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

[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 System Modeling*, 22(2):495–520, 2023. doi:10.1007/s10270-023-01092-7.

[DLR23] Elena Di Lavore and Mario Román. Evidential decision theory via partial markov categories. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–14, 2023. doi:10.1109/LICS56636.2023.10175776.

[EH97] Conal Elliott and Paul Hudak. Functional reactive animation. In *Proceedings of the second ACM SIGPLAN international conference on Functional*

programming, pages 263–273, 1997.

[ES22] Matthew Earnshaw and Paweł Sobociński. Regular Monoidal Languages. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.44.

[ES23] Matthew Earnshaw and Paweł Sobociński. String Diagrammatic Trace Theory. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.43.

[FL23] Tobias Fritz and Wendong Liang. Free gs-monoidal categories and free markov categories. *Applied Categorical Structures*, 31(2):21, 2023.

[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.

[GKS22] Dan R Ghica, George Kaye, and David Sprunger. A compositional theory of digital circuits, 2022. arXiv:2201.10456.

[HJS07] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Log. Methods Comput. Sci.*, 3(4), 2007. doi:10.2168/LMCS-3(4:11)2007.

[HMH14] Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In *Proceedings of the 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, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2603088.2603124.

[HPW98] Thomas Hildebrandt, Prakash Panangaden, and Glynn Winskel. A relational model of non-deterministic dataflow. In *CONCUR'98 Concurrency Theory: 9th International Conference Nice, France, September 8–11, 1998 Proceedings 9*, pages 613–628. Springer, 1998.

[Hug00] John Hughes. Generalising monads to arrows. *Sci. Comput. Program.*, 37(1-3):67–111, 2000. doi:10.1016/S0167-6423(99)00023-4.

[Jac10] Bart Jacobs. From coalgebraic to monoidal traces. *Electronic Notes in Theoretical Computer Science*, 264(2):125–140, 2010.

[Jac18] Bart Jacobs. From probability monads to commutative effectuses. *Journal of logical and algebraic methods in programming*, 94:200–237, 2018.

[Jef97] Alan Jeffrey. Premonoidal categories and a graphical view of programs. *Preprint at ResearchGate*, 1997. URL: https://www.researchgate.net/profile/Alan-Jeffrey/publication/228639836_Premonoidal_categories_and_a_graphical_view_of_programs/links/00b495182cd648a874000000/Premonoidal-categories-and-a-graphical-view-of-programs.pdf.

[JHH09] Bart Jacobs, Chris Heunen, and Ichiro Hasuo. Categorical semantics for arrows. *J. Funct. Program.*, 19(3-4):403–438, 2009. doi:10.1017/S0956796809007308.

[Jon89] Bengt Jonsson. A fully abstract trace model for dataflow networks. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 155–165, 1989.

[Kah74] Gilles Kahn. The semantics of a simple language for parallel programming. *Information processing*, 74(471-475):15–28, 1974.

[KBH12] Neelakantan R Krishnaswami, Nick Benton, and Jan Hoffmann. Higher-order functional reactive programming in bounded space. *ACM SIGPLAN Notices*, 47(1):45–58, 2012.

[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 FC Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115(2):141–178, 1997. doi:10.1016/S0022-4049(96)00012-6.

[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.

[Lev22] Paul Blain Levy. Call-by-Push-Value. *ACM SIGLOG News*, 9(2):7–29, may 2022. doi:10.1145/3537668.3537670.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991. doi:10.1016/0890-5401(91)90052-4.

[MS14] Rasmus Ejlers Møgelberg and Sam Staton. Linear usage of state. *Log. Methods Comput. Sci.*, 10(1), 2014. doi:10.2168/LMCS-10(1:17)2014.

[PS23] Hugo Paquet and Philip Saville. Effectful semantics in 2-dimensional categories: premonoidal and freyd bicategories. 2023. arXiv:2312.14964.

[PT97] John Power and Hayo Thielecke. Environments, continuation semantics and indexed categories. In *International Symposium on Theoretical Aspects of Computer Software*, pages 391–414. Springer, 1997.

[PW99] John Power and Hiroshi Watanabe. Distributivity for a monad and a comonad. In *CMCS*, page 102, 1999.

[PZ23] Robin Piedeleu and Fabio Zanasi. A finite axiomatisation of finite-state automata using string diagrams. *Logical Methods in Computer Science*, 19, 2023.

[Ran58] George N Raney. Sequential functions. *Journal of the ACM (JACM)*, 5(2):177–180, 1958.

[Rom22] Mario Román. Promonads and string diagrams for effectful categories. In Jade Master and Martha Lewis, editors, *Proceedings Fifth International Conference on Applied Category Theory, ACT 2022, Glasgow, United Kingdom, 18-22 July 2022*, volume 380 of *EPTCS*, pages 344–361, 2022. doi:10.4204/EPTCS.380.20.

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

[Rut95] Jan Rutten. A calculus of transition systems (towards universal coalgebra). In *Modal logic and process algebra : a bisimulation perspective*, pages 231–256, 01 1995.

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

[SBBR10] Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing the powerset construction, coalgebraically. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 272–283. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2010.272, doi:10.4230/LIPICS.FSTTCS.2010.272.

[SBBR13] Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9, 2013.

[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.

[Sta09] Sam Staton. Relating coalgebraic notions of bisimulation. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*, volume 5728 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2009. doi:10.1007/978-3-642-03741-2_14.

[Swe69] Moss Eisenberg Sweedler. Integrals for hopf algebras. *Annals of Mathematics*, pages 323–335, 1969.

[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.

[Voo20] Niels Frits Willem Voorneveld. *Equality between programs with effects: doctoral dissertation*. PhD thesis, Univerza v Ljubljani, Fakulteta za matematiko in fiziko, 2020.

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

A STRING DIAGRAMS

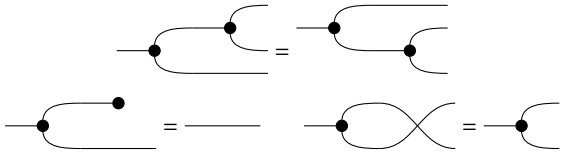


Figure 6: Axioms for a cocommutative comonoid.

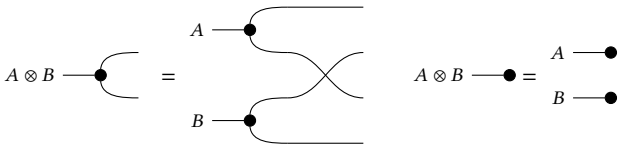


Figure 7: Axioms for a coherent copy-discard structure.

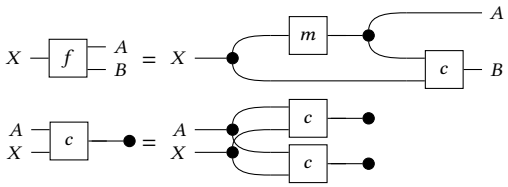


Figure 8: Conditionals and quasi-totality.

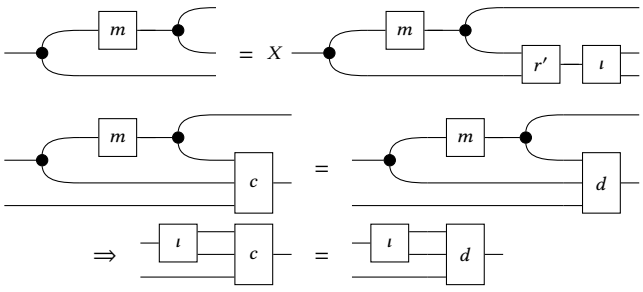


Figure 9: Axioms for ranges.

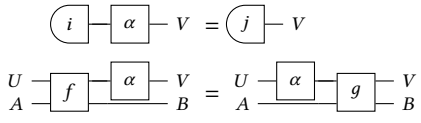


Figure 10: Morphisms of effectful Mealy machines.

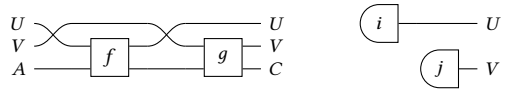


Figure 11: Composition of effectful Mealy machines.

B PROOFS ON DO-NOTATION

Definition B.1 (Graph homomorphism)**.** An *effectful copy-discard graph homomorphism*, $f\colon (O_1, \mathcal{V}_1, \mathcal{P}_1, C_1) \to (O_2, \mathcal{V}_2, \mathcal{P}_2, C_2)$, consists of (i) a function on objects, (ii) a function on value generators that translates their objects, (iii) a function on pure generators that translates their objects, and a (iv) function on effectful generators that translates their objects; that is,

(i) $f\colon O_1 \to O_2$;
(ii) $f_v\colon \mathcal{V}_1(X_1...X_n; Y) \to \mathcal{V}_2(fX_1...fX_n; fY)$;
(iii) $f_p\colon \mathcal{P}_1(X_1...X_n; Y_1...Y_m) \to \mathcal{P}_2(fX_1...fX_n; fY_1...fY_m)$;
(iv) $f_c\colon C_1(X_1...X_n; Y_1...Y_m) \to C_2(fX_1...fX_n; fY_1...fY_m)$.

Proposition B.2. *Interchanging the order of a pure statement with either a pure or an effectful statement on a well-formed derivation results on a well-formed derivation.*

Definition B.3 (Substitution)**.** Let $\Gamma \Vdash p : \mathbb{Y}$ and $\Gamma \vdash t : \mathbb{Y}$ be a derivation and a term, and let $(x_i : X_i) \in \Gamma$ be a variable. The substitution of the variable x_i by a term $s : X_i$, on a derivation or value, $p[s/x_i]$ or $t[s/x_i]$, is defined inductively over both derivations and values as follows.

(1) $x_i[s/x_i] = s$; and $x_j[s/x_i] = x_j$ when $j \neq i$;
(2) $\mathtt{return}(t_1, ..., t_n)[s/x_i] = \mathtt{return}(t_1[s/x_i], ..., t_n[s/x_i])$;
(3) $f(t_1, .., t_n)[s/x_i] = f(t_1[s/x_i], ..., t_n[s/x_i])$;
(4) $(g(t_1, ..., t_n) \to y_1, ..., y_n \fatsemi p)[s/x_i] = g(t_1[s/x_i], ..., t_n[s/x_i]) \to y_1, ..., y_n \fatsemi p[s/x_i]$;
(5) $(h(t_1, ..., t_n) \rightsquigarrow y_1, ..., y_n \fatsemi p)[s/x_i] = h(t_1[s/x_i], ..., t_n[s/x_i]) \rightsquigarrow y_1, ..., y_n \fatsemi p[s/x_i]$;

We must ensure that the variables that we create are fresh, different from x_i.

Definition B.4. The *composition* of two derivations

$$\Gamma \Vdash p : Y_1, ..., Y_n \quad \text{and} \quad y_1 : Y_1, ..., y_n : Y_n \vdash t : \mathbb{Z},$$

is the derivation $\Gamma \Vdash \mathrm{comp}(p; q) : \mathbb{Z}$ defined by structural induction as follows.

- $\mathrm{comp}(\mathtt{return}(t_1, ..., t_n); q) = q[t_1/y_1, ..., t_n/y_n]$;
- $\mathrm{comp}((g(t_1, ..., t_n) \to y_1, ..., y_n \fatsemi p); q) = g(t_1, ..., t_n) \to y_1, ..., y_n \fatsemi \mathrm{comp}(p; q)$;
- $\mathrm{comp}((h(t_1, ..., t_n) \rightsquigarrow y_1, ..., y_n \fatsemi p); q) = h(t_1, ..., t_n) \rightsquigarrow y_1, ..., y_n \fatsemi \mathrm{comp}(p; q)$.

The *identity* derivation simply gives back all the variables on a context, without changing their order or multiplicity,

$$x_1 : X_1, ..., x_n : X_n \vdash \mathtt{return}(x_1, ..., x_n) : X_1, ..., X_n.$$

Proposition B.5 (From Proposition 2.8)**.** *Do-notation derivations over an effectful copy-discard graph, $\mathcal{G} = (O, \mathcal{V}, \mathcal{P}, \mathcal{E})$, quotiented by interchange, form an effectful copy-discard category,* $\mathrm{Do}_{\mathcal{G}}(\mathcal{V}) \to \mathrm{Do}(\mathcal{P}) \to \mathrm{Do}(\mathcal{E})$.

Proof. The morphisms of the cartesian category, $\mathrm{Do}_{\mathcal{G}}(\mathcal{V})$, are derivations that only contain a return statement. The morphisms of the monoidal category, $\mathrm{Do}_{\mathcal{G}}(\mathcal{P})$, are derivations that only contain pure generators ($\to$) and values. The morphisms of the premonoidal category, $\mathrm{Do}_{\mathcal{G}}(\mathcal{E})$, are derivations nthat contain now values, pure generators, effectful generators ($\rightsquigarrow$).

Composition is defined as in the previous Definition B.4. Composition is associative and unital; composition also respects the property defining each one of the categories: the composition of return statements is a return statement; the composition of pure generator derivations is a pure generator derivation. □

C EFFECTFUL AND COPY-DISCARD CATEGORIES

Definition C.1. An *effectful copy-discard (strict) functor* between strict effectful copy-discard categories,

$$(F, G, H)\colon (\mathbf{V}, \mathbf{P}, \mathbf{C}) \to (\mathbf{V}', \mathbf{P}', \mathbf{C}'),$$

is a triple consisting of a two strict monoidal functors $F\colon \mathbf{V} \to \mathbf{V}'$ and $G\colon \mathbf{P} \to \mathbf{P}'$ and a functor $H\colon \mathbf{C} \to \mathbf{C}'$ strictly preserving the premonoidal structure. These functors must commute with the inclusions determining the effectful copy-discard categories, meaning that the following diagram must commute.

$$\begin{array}{ccccc} \mathbf{V} & \longrightarrow & \mathbf{P} & \longrightarrow & \mathbf{C} \\ {\scriptstyle F}\downarrow & & {\scriptstyle G}\downarrow & & {\scriptstyle H}\downarrow \\ V' & \longrightarrow & \mathbf{P}' & \longrightarrow & \mathbf{C}' \end{array}$$

Bayes inversions are a particular case of quasi-total conditionals. The *Bayesian inversion* of a morphism $g\colon Y \to Z$ with respect to $f\colon X \to Y$ is a morphism $g^*_f\colon X \otimes Z \to Y$ such that can recover the output of f from the output of $f \fatsemi g$.

$$\left.\begin{array}{l} f(x) \to y \\ g(y) \to z \\ g^*_f(x,z) \to m \\ \mathtt{return}\ (\mathtt{m},\mathtt{z}) \end{array}\right\} = \left\{\begin{array}{l} f(x) \to y \\ g(y) \to z \\ \mathtt{return}\ (\mathtt{y},\mathtt{z}) \end{array}\right\}$$

These equations correspond to asking that g^*_f is a quasi-total conditional of $f \fatsemi \sigma \fatsemi (g \otimes \mathrm{id})$.

D PROOFS ON EFFECTFUL STREAMS

Remark D.1 (Ehrenfest model). The Ehrenfest model represents a system with two urns and multiple particles; at each step, a randomly and uniformly picked particle jumps from its current urn to the other.

```
ehrenfest°() =
  yield ([],full)

ehrenfest+°(left,right) =
  uniform() → ball
  move(left,ball) → left
  move(right,ball) → right
  yield (left,right)

ehrenfest++ = ehrenfest+
```

Compare with the type-theoretic formalization of this same model in the work of Di Lavore, de Felice and Román, where feedback appears as a primitive operation [DLdFR22].

Lemma D.2. *The set of observational sequences from $\boldsymbol{X}$ to $\boldsymbol{Y}$ in a productive effectful category $(\mathbf{P}, \mathbf{C})$ is isomorphic to the set of effectful streams of the same type.*

Proof sketch. The proof proceeds analogous to the monoidal case [DLdFR22, DLdFR23]. □

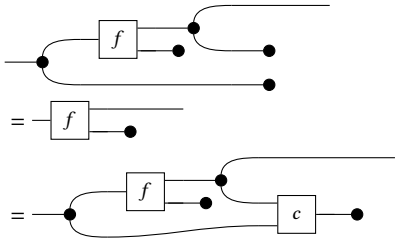
Corollary D.3. *Observational sequences over a productive effectful category $(\mathbf{P}, \mathbf{C})$ form an effectful category $\mathrm{OSeq}_{(\mathbf{P},\mathbf{C})}$, where objects are streams of objects $\boldsymbol{X} = (X_0, X_1, \dots)$ in $\mathbf{C}$.*

Proof. We define composition, identities and monoidal actions as in streams, using the bijection between hom-sets. □

Proof of Theorem 3.12. The bijections of Lemma D.2 define an effectful functor because the category of observational sequences is defined in order for these bijections to respect compositions, identities and monoidal actions. An identity-on-objects fully-faithful functor is an isomorphism. □

Lemma D.4. *Let $f : X \to A \otimes B$ be a morphism in a copy-discard category $\mathbf{C}$ with quasi-total conditionals and ranges. Then, all its quasi-total conditionals c are total on its range $r' \fatsemi \iota$, the composition $\iota \fatsemi c$ is total.*

Proof. We apply the properties of quasi-total conditionals.



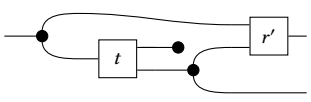
By the properties of ranges and totality of ι, we obtain the thesis, $\iota \fatsemi c \fatsemi \multimap = \iota \fatsemi \multimap = \multimap$. □

Definition D.5. For morphisms $m : X \to A$ and $c : X \otimes A \to B$ in a copy-discard category, define $m \triangleleft c : X \to B \otimes A$ as

$$m \triangleleft c := \left\{ \begin{array}{l} m(x) \to a \\ c(x,a) \to b \\ \texttt{return}\ \ (b,a) \end{array} \right\}$$

Proof of Proposition 3.13. Given a 1-stage process $\langle t|$, consider the range $r = r' \fatsemi \iota$ of $t \fatsemi (\multimap \otimes \mathrm{id})$ and define a candidate minimal representative $t_0 := (t \fatsemi (\multimap \otimes \mathrm{id})) \triangleleft r'$.

$$t_0 := \left\{ \begin{array}{l} t(x) \to (m,y) \\ r'(x,y) \to s \\ \texttt{return}\ \ (s) \end{array} \right\}$$



This is an element of $\langle t|$ because we can rewrite $\langle t|$ as below applying the properties of a quasi-total conditional c of t, the properties of the range r, and Lemma D.4.

$$\begin{aligned} &\langle t| \\ &= \langle (t \fatsemi (\multimap \otimes \mathrm{id})) \triangleleft c| \\ &= \left\langle (t \fatsemi (\multimap \otimes \mathrm{id})) \triangleleft (r' \fatsemi \iota \fatsemi c) \right| \\ &\sim \left\langle (t \fatsemi (\multimap \otimes \mathrm{id})) \triangleleft r' \right| \\ &= \langle t_0| \end{aligned}$$

We show that t_0 does not depend on the particular representative t. Suppose that $\langle t| = \langle u|$ in k sliding steps and proceed by induction on k. If $k = 0$, then $t = u$ and we are done. Otherwise, suppose that there is v such that $\langle u| = \langle v|$ in $k-1$ steps and there are total morphisms a and b such that $v \fatsemi (a \otimes \mathrm{id}) = t \fatsemi (b \otimes \mathrm{id})$. Since a and b are total, we obtain that $v \fatsemi (\multimap \otimes \mathrm{id}) = t \fatsemi (\multimap \otimes \mathrm{id})$. By induction hypothesis, $u \fatsemi (\multimap \otimes \mathrm{id}) = v \fatsemi (\multimap \otimes \mathrm{id})$, which gives $t \fatsemi (\multimap \otimes \mathrm{id}) = u \fatsemi (\multimap \otimes \mathrm{id})$ and that t_0 does not depend on the chosen representative t.

We define a candidate total morphism s_t for every element $t \in \langle t_0|$ as one of its quasi-total conditionals precomposed with ι: $s_t := \iota \fatsemi c_t$. By the definition of conditionals and ranges, $t_0 \fatsemi (s_t \otimes \mathrm{id}) = t$.

We show that they satisfy the last condition for productivity by the axioms of ranges. Suppose that $t, u \in \langle t_0|$ and

$$\left\langle \left\{ \begin{array}{l} t(x) \to (m,y) \\ a(x') \to (m',y') \\ \texttt{return}\ \ (m',y,y') \end{array} \right\} \right| = \left\langle \left\{ \begin{array}{l} u(x) \to (m,y) \\ b(x') \to (m',y') \\ \texttt{return}\ \ (m',y,y') \end{array} \right\} \right| .$$

By the first point in this proof, $t \fatsemi (\multimap \otimes \mathrm{id}) = u \fatsemi (\multimap \otimes \mathrm{id}) = h$ and

$$\left\{ \begin{array}{l} t(x) \to (m,y) \\ a(m,x') \to (m',y') \\ \texttt{return}\ \ (y,y') \end{array} \right\} = \left\{ \begin{array}{l} u(x) \to (m,y) \\ b(m,x') \to (m',y') \\ \texttt{return}\ \ (y,y') \end{array} \right\} .$$

This equation implies, for c and d quasi-total conditionals of t and u, that

$$\left\{ \begin{array}{l} h(x) \to y \\ c(x,y) \to m \\ a(m,x') \to (m',y') \\ \texttt{return}\ \ (y,y') \end{array} \right\} = \left\{ \begin{array}{l} h(x) \to y \\ d(x,y) \to m \\ b(m,x') \to (m',y') \\ \texttt{return}\ \ (y,y') \end{array} \right\} .$$

By the properties of ranges, we obtain that

$$((\iota \fatsemi c) \otimes \mathrm{id}) \fatsemi a \fatsemi (\multimap \otimes \mathrm{id}) = ((\iota \fatsemi d) \otimes \mathrm{id}) \fatsemi b \fatsemi (\multimap \otimes \mathrm{id}) .$$

We apply the sliding equivalence and obtain

$$\langle ((\iota \fatsemi c) \otimes \mathrm{id}) \fatsemi a| \sim \langle ((\iota \fatsemi d) \otimes \mathrm{id}) \fatsemi b| .$$

□

E PROOFS ON CAUSAL PROCESSES

Proof of Proposition 4.2. We show that causal processes are a category where identities and composition are defined componentwise by those in $\mathbf{C}$. Since they are defined component wise, they must be associative and unital. We proceed to check that they are well-defined, i.e. that the identity satisfies the causality condition and that, whenever f and g satisfy the causal condition, $f \fatsemi g$ does so too. The projection $\pi_{\cdot} : A_0 \otimes \cdots \otimes A_{n+1} \to A_{n+1}$ on the last coordinate is always a conditional of the identity $\mathrm{id}_{A_0 \otimes \cdots \otimes A_{n+1}}$ with respect to the marginal $\mathrm{id}_{A_0 \otimes \cdots \otimes A_n}$, which shows that the causality condition is satisfied. For compositions, suppose that the processes $\boldsymbol{f} : \boldsymbol{A} \to \boldsymbol{B}$ and $\boldsymbol{g} : \boldsymbol{B} \to \mathit{streamC}$ have conditionals $c_n : B_0 \otimes \cdots \otimes B_{n-1} \otimes A_0 \otimes \cdots \otimes A_n \to B_n$ and $d_n : C_0 \otimes \cdots \otimes C_{n-1} \otimes B_0 \otimes \cdots \otimes B_n \to C_n$. We show that $f_n \fatsemi g_n$ is a marginal of $f_{n+1} \fatsemi g_{n+1}$ with do-notation.

$$\begin{aligned} &(\boldsymbol{f} \fatsemi \boldsymbol{g})_{n+1} \\ &:= f_{n+1} \fatsemi g_{n+1} \end{aligned}$$

$$
\left.\begin{array}{l} f_n(\vec{x}_n) \to \vec{y}_n \\ c_{n+1}(\vec{y}_n, \vec{x}_{n+1}) \to y_{n+1} \\ = g_n(\vec{y}_n) \to \vec{z}_n \\ d_{n+1}(\vec{z}_n, \vec{y}_{n+1}) \to z_{n+1} \\ \texttt{return } (\vec{z}_{n+1}) \end{array}\right\}
$$

$$
\left.\begin{array}{l} f_n(\vec{x}_n) \to \vec{y}_n \\ g_n(\vec{y}_n) \to \vec{z}_n \\ = c_{n+1}(\vec{y}_n, \vec{x}_{n+1}) \to y_{n+1} \\ d_{n+1}(\vec{z}_n, \vec{y}_{n+1}) \to z_{n+1} \\ \texttt{return } (\vec{z}_{n+1}) \end{array}\right\} \tag{2}
$$

$$
\left.\begin{array}{l} f_n(\vec{x}_n) \to \vec{m}_n \\ g_n(\vec{m}_n) \to \vec{z}_n \\ = g_f^*(\vec{x}_n, \vec{z}_n) \to \vec{y}_n \\ c_{n+1}(\vec{y}_n, \vec{x}_{n+1}) \to y_{n+1} \\ d_{n+1}(\vec{z}_n, \vec{y}_{n+1}) \to z_{n+1} \\ \texttt{return } (\vec{z}_{n+1}) \end{array}\right\} \tag{3}
$$

$$
\left.\begin{array}{l} (f \fatsemi g)_n(\vec{x}_n) \to \vec{z}_n \\ g_f^*(\vec{x}_n, \vec{z}_n) \to \vec{y}_n \\ = c_{n+1}(\vec{y}_n, \vec{x}_{n+1}) \to y_{n+1} \\ d_{n+1}(\vec{z}_n, \vec{y}_{n+1}) \to z_{n+1} \\ \texttt{return } (\vec{z}_{n+1}) \end{array}\right\}
$$

The category $\mathbf{C}$ is monoidal so statements can be interchanged (Equation (2)). Equation (3) uses the Bayes inversion g_f^* of g_n with respect to f_n.

The monoidal product $\boldsymbol{f \otimes g}$ is defined on the components by reshuffling the order of the inputs and outputs of the monoidal product $f_n \otimes g_n$ in $\mathbf{C}$.

$$
(f \otimes g)_n \coloneqq \left\{\begin{array}{l} f_n(\vec{w}_n) \to \vec{y}_n \\ g_n(\vec{x}_n) \to \vec{z}_n \\ \texttt{return } (\overrightarrow{(y,z)}_n) \end{array}\right\}
$$

This operation preserves identities and compositions.

$$
\begin{aligned}
&(\mathrm{id}_{\boldsymbol{A}} \otimes \mathrm{id}_{\boldsymbol{B}})_n \\
&\quad \left.\begin{array}{l} \mathrm{id}_{\vec{A}_n}(\vec{a}_n) \to \vec{a'}_n \\ \coloneqq \mathrm{id}_{\vec{B}_n}(\vec{b}_n) \to \vec{b'}_n \\ \texttt{return } (\overrightarrow{(a',b')}_n) \end{array}\right\} \\
&\quad = \texttt{return } (\overrightarrow{(a,b)}_n) \quad \Big\} \\
&\quad = (\mathrm{id}_{\boldsymbol{A}\otimes\boldsymbol{B}})_n
\end{aligned}
$$

$$
\begin{aligned}
&((f \otimes f') \fatsemi (g \otimes g'))_n \\
&\quad \left.\begin{array}{l} f_n(\vec{x}_n) \to \vec{y}_n \\ f'_n(\vec{x'}_n) \to \vec{y'}_n \\ \coloneqq g_n(\vec{y}_n) \to \vec{z}_n \\ g'_n(\vec{y'}_n) \to \vec{z'}_n \\ \texttt{return } (\overrightarrow{(z,z')}_n) \end{array}\right\} \\
&\quad \left.\begin{array}{l} f_n(\vec{x}_n) \to \vec{y}_n \\ g_n(\vec{y}_n) \to \vec{z}_n \\ = f'_n(\vec{x'}_n) \to \vec{y'}_n \\ g'_n(\vec{y'}_n) \to \vec{z'}_n \\ \texttt{return } (\overrightarrow{(z,z')}_n) \end{array}\right\} \\
&\quad = ((f \fatsemi g) \otimes (f' \fatsemi g'))_n
\end{aligned} \tag{4}
$$

Finally, the monoidal product is well-defined because $(f \otimes g)_n$ is a marginal for $(f \otimes g)_{n+1}$.

$$
\begin{aligned}
&(f \otimes g)_{n+1} \\
&\quad \left.\begin{array}{l} f_{n+1}(\vec{w}_{n+1}) \to \vec{y}_{n+1} \\ \coloneqq g_{n+1}(\vec{x}_{n+1}) \to \vec{z}_{n+1} \\ \texttt{return } (\overrightarrow{(y,z)}_{n+1}) \end{array}\right\} \\
&\quad \left.\begin{array}{l} f_n(\vec{w}_n) \to \vec{y}_n \\ c_{n+1}(\vec{w}_{n+1}, \vec{y}_n) \to y_{n+1} \\ = g_n(\vec{x}_n) \to \vec{z}_n \\ d_{n+1}(\vec{x}_{n+1}, \vec{z}_n) \to z_{n+1} \\ \texttt{return } (\overrightarrow{(y,z)}_{n+1}) \end{array}\right\} \qquad (5) \\
&\quad \left.\begin{array}{l} f_n(\vec{w}_n) \to \vec{y}_n \\ g_n(\vec{x}_n) \to \vec{z}_n \\ = c_{n+1}(\vec{w}_{n+1}, \vec{y}_n) \to y_{n+1} \\ d_{n+1}(\vec{x}_{n+1}, \vec{z}_n) \to z_{n+1} \\ \texttt{return } (\overrightarrow{(y,z)}_{n+1}) \end{array}\right\} \qquad (6) \\
&\quad = \left\{\begin{array}{l} (f \otimes g)_n(\overrightarrow{(w,x)}_n) \to \overrightarrow{(y,z)}_n \\ c_{n+1}(\vec{w}_{n+1}, \vec{y}_n) \to y_{n+1} \\ d_{n+1}(\vec{x}_{n+1}, \vec{z}_n) \to z_{n+1} \\ \texttt{return } (\overrightarrow{(y,z)}_{n+1}) \end{array}\right\}
\end{aligned}
$$

Equation (5) applies the causality condition to f_{n+1} and g_{n+1}, and Equations (4) and (6) use interchange. □

Lemma E.1. *Consider two observational sequences $\langle t_n \mid n \in \mathbb{N}\rangle$ and $\langle u_n \mid n \in \mathbb{N}\rangle$, and suppose there are a_n pure such that $(a_{n-1} \otimes \mathrm{id}) \fatsemi t_n = u_n \fatsemi (a_n \otimes \mathrm{id})$. Then $\langle t_n \mid n \in \mathbb{N}\rangle = \langle u_n \mid n \in \mathbb{N}\rangle$.*

Proof. Show by induction on n that

$$
\langle t_0 \mid \cdots \mid t_n| = \langle u_0 \mid \cdots \mid u_n \fatsemi (a_n \otimes \mathrm{id})|
$$

by sliding only identities on the last memory. If $n = 0$ then $t_0 = u_0 \fatsemi (a_0 \otimes \mathrm{id})$ by hypothesis. Suppose the thesis is true for n.

$$
\begin{aligned}
&\langle t_0 \mid \cdots \mid t_n \mid t_{n+1}| \\
&\sim \langle u_0 \mid \cdots \mid u_n \fatsemi (a_n \otimes \mathrm{id}) \mid t_{n+1}| \\
&\sim \langle u_0 \mid \cdots \mid u_n \mid (a_n \otimes \mathrm{id}) \fatsemi t_{n+1}| \\
&= \langle u_0 \mid \cdots \mid u_n \mid u_{n+1} \fatsemi (a_{n+1} \otimes \mathrm{id})|
\end{aligned}
$$

With this, we show that, for every $n \in \mathbb{N}$,

$$
\begin{aligned}
&\langle t_0 \mid \cdots \mid t_n| \\
&\sim \langle u_0 \mid \cdots \mid u_n \fatsemi (a_n \otimes \mathrm{id})| \\
&\sim \langle u_0 \mid \cdots \mid u_n|
\end{aligned}
$$

□

Lemma E.2. *Let $c_n \colon \vec{A}_{n-1} \otimes \vec{B}_{n-1} \otimes A_n \to B_n$ be a family of morphisms, for $n \in \mathbb{N}$, and define r'_0, j_0 to be the range of c_0 and r'_{n+1}, j_{n+1} to be the range of $(j_n \otimes \mathrm{id}) \fatsemi c_{n+1}$. Then,*

$$
\left\langle \left\{\begin{array}{l} c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{b}_n, \vec{a}_n, b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle
$$

$$
\sim \left\langle \left\{\begin{array}{l} j_{n-1}(s) \to (\vec{a}_{n-1}, \vec{b}_{n-1}) \\ c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ r'_n(\vec{a}_n, \vec{b}_n) \to s' \\ \texttt{return } (s', b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle
$$

Proof. By determinism of j_n and their definition, we have the following equalities.

$$\left.\begin{array}{l} j_{n-1}(s) \to (\vec{a}_{n-1}, \vec{b}_{n-1}) \\ c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{a}_{n-1}, \vec{b}_{n-1}, a_n, b_n, b_n) \end{array}\right\} = \left\{\begin{array}{l} c_n(j_{n-1}(s), a_n) \to b_n \\ \texttt{return } (j_{n-1}(s), a_n, b_n, b_n) \end{array}\right\}$$

$$= \left\{\begin{array}{l} c_n(j_{n-1}(s), a_n) \to b_n \\ r_n(s, a_n, b_n) \to (s', a'_n, b'_n) \\ \texttt{return } (j_{n-1}(s'), a'_n, b'_n, b_n) \end{array}\right\}$$

$$= \left\{\begin{array}{l} c_n(j_{n-1}(s), a_n) \to b_n \\ r'_n(s, a_n, b_n) \to s' \\ \texttt{return } (j_n(s'), b_n) \end{array}\right\}$$

We can, then, apply Lemma E.1 with $a_n := j_n$ and get the thesis. □

Lemma E.3. *Consider two families of conditionals*

$$c_n, d_n \colon \vec{A}_{n-1} \otimes \vec{B}_{n-1} \otimes A_n \to B_n$$

that induce the same family of morphisms,

$$\left.\begin{array}{l} c_0(a_0) \to b_0 \\ \vdots \\ c_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\} = \left\{\begin{array}{l} d_0(a_0) \to b_0 \\ \vdots \\ d_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\}.$$

Then they are equal up to appropriate ranges,

$$c_n(j_{n-1}(s_{n-1}), a_n) = d_n(j_{n-1}(s_{n-1}), a_n)\,,$$

where $j_0 = \iota_0 \colon R_0 \to A_0 \otimes B_0$ is the total part of the range $r_0 = r'_0 \fatsemi \iota_0$ of c_0, ι_{n+1} is the total part of the range of $(j_n \otimes \mathrm{id}) \fatsemi c_{n+1}$, and $j_{n+1} = \iota_{n+1} \fatsemi (j_n \otimes \mathrm{id})$.

Proof. Proceed by induction. The base case checks that $c_0 = d_0$, by hypothesis. For the inductive step, we need an auxiliary inductive lemma to show that, whenever

$$\left.\begin{array}{l} c_k(\vec{a}_{k-1}, \vec{b}_{k-1}, a_k) \to b_k \\ \vdots \\ c_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\} = \left\{\begin{array}{l} d_k(\vec{a}_{k-1}, \vec{b}_{k-1}, a_k) \to b_k \\ \vdots \\ d_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\},$$

and $c_k(j_{k-1}(s_{k-1}), a_k) = d_k(j_{k-1}(s_{k-1}), a_k)$, then we can deduce, by the properties of ranges, the definition of j_k and the inductive hypothesis on k, that

$$\left.\begin{array}{l} c_{k+1}(\vec{a}_k, \vec{b}_k, a_{k+1}) \to b_{k+1} \\ \vdots \\ c_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\} = \left\{\begin{array}{l} d_{k+1}(\vec{a}_k, \vec{b}_k, a_{k+1}) \to b_{k+1} \\ \vdots \\ d_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\}.$$

With this auxiliary lemma, the initial hypothesis and the induction hypothesis, i.e. that $c_k(j_{k-1}(s_{k-1}), a_k) = d_k(j_{k-1}(s_{k-1}), a_k)$ for $k \le n$, we conclude that $c_{n+1}(j_n(s_n), a_{n+1}) = d_{n+1}(j_n(s_n), a_{n+1})$. □

Lemma E.4. *For any two equivalent observational sequences, $\langle t_n \mid n \in \mathbb{N}\rangle \sim \langle u_n \mid n \in \mathbb{N}\rangle$, all their n-th executions coincide.*

$$\left.\begin{array}{l} t_0(a_0) \to m_0, b_0 \\ \vdots \\ t_n(m_{n-1}, a_n) \to m_n, b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\} = \left\{\begin{array}{l} u_0(a_0) \to m_0, b_0 \\ \vdots \\ u_n(m_{n-1}, a_n) \to m_n, b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\}$$

Proof. This is easily shown by induction using the fact that sliding is on total maps. □

Lemma E.5. *For two causal processes $(f_n \mid n \in \mathbb{N}) : \boldsymbol{X} \to \boldsymbol{Y}$ and $(g_n \mid n \in \mathbb{N}) : \boldsymbol{Y} \to \boldsymbol{Z}$,*

$$\left\langle \left\{\begin{array}{l} c_n(\vec{x}_{n-1}, \vec{y}_{n-1}, x_n) \to y_n \\ d_n(\vec{y}_{n-1}, \vec{z}_{n-1}, y_n) \to z_n \\ \texttt{return } (\vec{x}_n, \vec{y}_{n-1}, y_n, \vec{y}'_{n-1}, y_n, \vec{z}_n, z_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle$$

$$= \left\langle \left\{\begin{array}{l} g^*_{f_{n-1}}(\vec{x}_{n-1}, \vec{z}_{n-1}) \to y_n \\ c_n(\vec{x}_{n-1}, \vec{y}_{n-1}, x_n) \to y_n \\ d_n(\vec{y}_{n-1}, \vec{z}_{n-1}, y_n) \to z_n \\ \texttt{return } (\vec{x}_n, \vec{z}_n, z_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle$$

Proof. Let $v_n, w_n \colon \vec{X}_{n-1} \otimes \vec{Z}_{n-1} \otimes X_n \to Z_n$ be defined by

$$\begin{array}{l} v_n(\vec{x}_{n-1}, \vec{z}_{n-1}, x_n) = \\ \quad g^*_{f_{n-1}}(\vec{x}_{n-1}, \vec{z}_{n-1}) \rightsquigarrow \vec{y}_{n-1} \\ \quad c_n(\vec{x}_{n-1}, \vec{y}_{n-1}, x_n) \rightsquigarrow y_n \\ \quad d_n(\vec{y}_{n-1}, \vec{z}_{n-1}, y_n) \rightsquigarrow z_n \\ \quad \texttt{return } (z_n) \end{array} \qquad \begin{array}{l} v_n(\vec{x}_{n-1}, \vec{z}_{n-1}, x_n) = \\ \quad g^*_{f_{n-1}}(\vec{x}_{n-1}, \vec{z}_{n-1}) \rightsquigarrow \vec{y}_{n-1} \\ \quad c_n(\vec{x}_{n-1}, \vec{y}_{n-1}, x_n) \rightsquigarrow y_n \\ \quad d_n(\vec{y}_{n-1}, \vec{z}_{n-1}, y_n) \rightsquigarrow z_n \\ \quad g^*_{f_n}(\vec{x}_n, \vec{z}_n) \rightsquigarrow \vec{y}'_n \\ \quad \texttt{return } (z_n) \end{array}$$

We can check that $v_n \triangleleft \mathrm{id}$ and $w_n \triangleleft \mathrm{id}$ define the same family of morphisms, $(f_n \fatsemi g_n \mid n \in \mathbb{N})$, as in Lemma E.3. Applying Lemma E.3, we obtain that $(j_{n-1} \otimes \mathrm{id}) \fatsemi v_n = (j_{n-1} \otimes \mathrm{id}) \fatsemi w_n$. Lemma E.2 implies that $\langle v_n \triangleleft \mathrm{id} \mid n \in \mathbb{N}\rangle \sim \left\langle ((j_{n-1} \otimes \mathrm{id}) \fatsemi v_n) \triangleleft r'_n \mid n \in \mathbb{N}\right\rangle$ and $\langle w_n \triangleleft \mathrm{id} \mid n \in \mathbb{N}\rangle \sim \left\langle ((j_{n-1} \otimes \mathrm{id}) \fatsemi w_n) \triangleleft r'_n \mid n \in \mathbb{N}\right\rangle$. From these two facts, we deduce that $\langle v_n \triangleleft \mathrm{id} \mid n \in \mathbb{N}\rangle \sim \langle w_n \triangleleft \mathrm{id} \mid n \in \mathbb{N}\rangle$. Finally, we apply Lemma E.1 with the total morphism a_n defined below.

$$\begin{array}{l} a_n(s_n) = \\ \quad j_{n-1}(s_n) \rightsquigarrow \vec{x}_n, \vec{z}_n \\ \quad g^*_{f_n}(\vec{x}_n, \vec{z}_n) \rightsquigarrow \vec{y}_n \\ \quad \texttt{return } (\vec{x}_n, \vec{y}_n, \vec{y}_n, \vec{z}_n) \end{array}$$

□

Remark E.6. The sequence $(f_n \mid n \in \mathbb{N})$ defined in Proposition 4.4 is, more explicitly, defined by induction via the auxiliary morphisms $\hat{f}_n \colon X_0 \otimes \cdots \otimes X_n \to M_n \otimes Y_0 \otimes \cdots \otimes Y_n$.

$$\hat{f}_0 := t_0; \quad \hat{f}_{n+1} := \left\{\begin{array}{l} \hat{f}_n(\vec{x}_n) \to (m', \vec{y}_n) \\ t_{n+1}(m', x_{n+1}) \to (m, y_{n+1}) \\ \texttt{return } (m, \vec{y}_{n+1}) \end{array}\right\}$$

The components f_n are obtained from $\hat{f}_n$ by projecting out the memory, $f_n := \hat{f}_n \fatsemi \pi_{\vec{Y}_n}$.

Proof of Proposition 4.3. For a morphism $\boldsymbol{f} \colon \boldsymbol{A} \to \boldsymbol{B}$ in Causal$(\mathbf{C})$, consider the conditionals $c_n \colon \vec{B}_{n-1} \otimes \vec{A}_n \to B_n$ of f_n with respect to f_{n-1}, and, for a morphism $\langle t_n \mid n \in \mathbb{N}\rangle : \boldsymbol{A} \to \boldsymbol{B}$ in $\mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}$,

consider its n-th executions, $\hat{g}_n \colon \vec{A}_n \to M_n \otimes \vec{B}_n$ defined by induction.

$$\begin{aligned} \hat{g}_0 &\coloneqq t_0 \\ \hat{g}_{n+1} &\coloneqq \left.\begin{array}{l} \hat{g}_n(\vec{a}_n) \to (s', \vec{b}_n) \\ t_{n+1}(s', a_{n+1}) \to (s, b_{n+1}) \\ \texttt{return } (s, \vec{b}_{n+1}) \end{array}\right\} \end{aligned}$$

With these, we define the candidate functors $\mathrm{obs} \colon \mathrm{Causal}(\mathbf{C}) \to \mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}$ and $\mathrm{proc} \colon \mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})} \to \mathrm{Causal}(\mathbf{C})$ as identity on objects and, on morphisms using the notation above.

$$\mathrm{obs}(f) \coloneqq \left\langle \left.\begin{array}{l} c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{a}_n, \vec{b}_n, b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle$$

$$\mathrm{proc}(\langle t_n \mid n \in \mathbb{N}\rangle) \coloneqq \left(\left.\begin{array}{l} \hat{g}_n(\vec{a}_n) \to (s, \vec{b}_n) \\ \texttt{return } (\vec{b}_n) \end{array}\right\} \mid n \in \mathbb{N} \right)$$

We show that obs is well-defined, i.e. that any two choices c_n and d_n of conditionals for f_n define the same observational sequence. We apply Lemmas E.2 and E.3.

$$\begin{aligned} &\left\langle \left.\begin{array}{l} c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{b}_n, \vec{a}_n, b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle \\ &\sim \left\langle \left.\begin{array}{l} j_{n-1}(s) \to (\vec{a}_{n-1}, \vec{b}_{n-1}) \\ c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ r'_n(\vec{a}_n, \vec{b}_n) \to s' \\ \texttt{return } (s', b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle \\ &= \left\langle \left.\begin{array}{l} j_{n-1}(s) \to (\vec{a}_{n-1}, \vec{b}_{n-1}) \\ d_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ r'_n(\vec{a}_n, \vec{b}_n) \to s' \\ \texttt{return } (s', b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle \\ &\sim \left\langle \left.\begin{array}{l} c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{b}_n, \vec{a}_n, b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle \end{aligned}$$

We show that obs preserves identities. The identity causal process has identities as components. The projection π_{X_n} is a conditional of $\mathrm{id}_{\vec{X}_n}$ with respect to the marginal $\mathrm{id}_{\vec{X}_{n-1}}$. Then, $\mathrm{obs}(\mathrm{id}_X) = \langle \pi_n \triangleleft \mathrm{id} \mid n \in \mathbb{N}\rangle$. We can rewrite this sequence to

$$\left\langle \mathrm{id} \otimes (\multimap\!\bullet \fatsemi (\multimap\!\bullet \otimes \mathrm{id})) \mid n \in \mathbb{N} \right\rangle ,$$

which, applying Lemma E.1 with $u_n = \multimap\!\bullet \otimes \mathrm{id}$, is equivalent to $\left\langle \mathrm{id}_{X_n} \mid n \in \mathbb{N} \right\rangle$. This is the identity observational sequence.

The mapping obs preserves composition by Lemma E.5 and is monoidal by applying Lemma E.1 with the symmetries. By Lemma E.4, proc is well-defined. □

Lemma E.7. *For a copy-discard category* $\mathbf{C}$ *with quasi-total conditionals and ranges, the functor defined in Proposition 4.3 is faithful.*

Proof. We show that, for every X and Y, the hom-set $\mathrm{Causal}(\mathbf{C})(X, Y)$ is included in $\mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}(X, Y)$ via obs with inverse proc. We check that $\mathrm{proc}(\mathrm{obs}(f)) = f$ for every $f \colon X \to Y$ in $\mathrm{Causal}(\mathbf{C})$.

$$\begin{aligned} &\mathrm{proc}(\mathrm{obs}(f)) \\ &\coloneqq \mathrm{proc}\left(\left\langle \left.\begin{array}{l} c_n(\vec{a}_n, \vec{b}_{n-1}) \to b_n \\ \texttt{return } (\vec{a}_n, \vec{b}_n, b_n) \end{array}\right\} \mid n \in \mathbb{N} \right\rangle \right) \\ &\coloneqq \left(\left.\begin{array}{l} c_0(a_0) \to b_0 \\ \vdots \\ c_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{b}_n) \end{array}\right\} \right) \\ &= f \end{aligned}$$

The last equality holds because c_n is defined as the conditional of f_n with respect to the marginal $f_{n-1} \otimes \multimap\!\bullet$. □

Lemma E.8. *Let* $\langle t_n \mid n \in \mathbb{N}\rangle$ *be an observational sequence, and consider* $(f_n \mid n \in \mathbb{N}) = \mathrm{proc}\langle t_n \mid n \in \mathbb{N}\rangle$, $q_{-1} \coloneqq \mathrm{id}_I$ *and* q_n *to be a quasi-total conditional of* $(q_{n-1} \otimes \mathrm{id}) \fatsemi t_n$. *Then,* $(q_{n-1} \otimes \mathrm{id}) \fatsemi t_n \fatsemi (\multimap\!\bullet \otimes \mathrm{id})$ *is a conditional of* f_n *with respect to the marginal* f_{n-1}.

Proof. Proceed by induction on n with the stronger induction hypothesis below, where $\hat{f}_n$ is defined as in the proof of Proposition 4.4.

$$\hat{f}_n = \left.\begin{array}{l} f_{n-1}(\vec{a}_{n-1}) \to \vec{b}_{n-1} \\ q_{n-1}(\vec{a}_{n-1}, \vec{b}_{n-1}) \to m' \\ t_n(m', a_n) \to (m, b_n) \\ \texttt{return } (m, \vec{b}_n) \end{array}\right\}$$

If $n = 0$, then $\hat{f}_0 \coloneqq t_0$ and we are done. Suppose that the statement is true for n, and we compute $\hat{f}_{n+1}$.

$$\begin{aligned} &\hat{f}_{n+1} \\ &\coloneqq \left.\begin{array}{l} \hat{f}_n(\vec{a}_n) \to (m', \vec{b}_n) \\ t_{n+1}(m', a_{n+1}) \to (m, b_{n+1}) \\ \texttt{return } (m, \vec{b}_{n+1}) \end{array}\right\} \\ &= \left.\begin{array}{l} f_{n-1}(\vec{a}_{n-1}) \to \vec{b}_{n-1} \\ q_{n-1}(\vec{a}_{n-1}, \vec{b}_{n-1}) \to m' \\ t_n(m', a_n) \to (m, b_n) \\ t_{n+1}(m', a_{n+1}) \to (m, b_{n+1}) \\ \texttt{return } (m, \vec{b}_{n+1}) \end{array}\right\} \\ &= \left.\begin{array}{l} f_{n-1}(\vec{a}_{n-1}) \to \vec{b}_{n-1} \\ q_{n-1}(\vec{a}_{n-1}, \vec{b}_{n-1}) \to m' \\ t_n(m', a_n) \to (\vec{m}, b_n) \\ q_n(\vec{a}_n, \vec{b}_n) \to m \\ t_{n+1}(m', a_{n+1}) \to (m, b_{n+1}) \\ \texttt{return } (m, \vec{b}_{n+1}) \end{array}\right\} \\ &= \left.\begin{array}{l} \hat{f}_n(\vec{a}_n) \to m', \vec{b}_n \\ q_n(\vec{a}_n, \vec{b}_n) \to m \\ t_{n+1}(m', a_{n+1}) \to (m, b_{n+1}) \\ \texttt{return } (m, \vec{b}_{n+1}) \end{array}\right\} \\ &= \left.\begin{array}{l} f_n(\vec{a}_n) \to \vec{b}_n \\ q_n(\vec{a}_n, \vec{b}_n) \to m \\ t_{n+1}(m', a_{n+1}) \to (m, b_{n+1}) \\ \texttt{return } (m, \vec{b}_{n+1}) \end{array}\right\} \end{aligned}$$

The thesis follows from this by discarding the memory produced. □

Lemma E.9. *For a copy-discard category* $\mathbf{C}$ *with quasi-total conditionals and ranges, the functor defined in Proposition 4.3 is full.*

Proof. We show that, for every X and Y, the hom-set $\mathrm{Causal}(\mathbf{C})(X,Y)$ maps surjectively in $\mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}(X,Y)$ via obs and proc. We compute $\mathrm{obs}(\mathrm{proc}(\langle t_n \mid n \in \mathbb{N}\rangle))$ and show by induction that this is equivalent to $\langle t_n \mid n \in \mathbb{N}\rangle$ for every $\langle t_n \mid n \in \mathbb{N}\rangle : X \to Y$ in $\mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}$.

Define the families of morphisms $\hat{f}_n$ and f_n.

$$\hat{f}_n := \left\{ \begin{array}{l} t_0(x_0) \to (m_0, y_0) \\ \vdots \\ t_n(m_{n-1}, x_n) \to (m_n, y_n) \\ \texttt{return } (m_n, \vec{y}_n) \end{array} \right\} \qquad f_n := \left\{ \begin{array}{l} \hat{f}_n(\vec{x}_n) \to (m_n, \vec{y}_n) \\ \texttt{return } (\vec{y}_n) \end{array} \right\}$$

By definition of proc, $(f_n \mid n \in \mathbb{N}) = \mathrm{proc}\langle t_n \mid n \in \mathbb{N}\rangle$. Let $q_{-1} := \mathrm{id}_I$ and q_n be a quasi-total conditional of $(q_{n-1} \otimes \mathrm{id}) \fatsemi t_n$. By Lemma E.8, $c_n := (q_{n-1} \otimes \mathrm{id}) \fatsemi t_n \fatsemi (\multimap \otimes \mathrm{id})$ is a conditional of f_n with respect to the marginal f_{n-1}. Then, the family c_n gives the image of $\langle t_n \mid n \in \mathbb{N}\rangle$.

$$\begin{aligned} &\mathrm{obs}(\mathrm{proc}(\langle t_n \mid n \in \mathbb{N}\rangle)) \\ &= \mathrm{obs}((f_n \mid n \in \mathbb{N})) \\ &= \left\langle \left\{ \begin{array}{l} c_n(\vec{a}_{n-1}, \vec{b}_{n-1}, a_n) \to b_n \\ \texttt{return } (\vec{a}_n, \vec{b}_n, b_n) \end{array} \right\} \mid n \in \mathbb{N} \right\rangle \\ &=: \langle u_n \mid n \in \mathbb{N}\rangle \end{aligned}$$

We show by induction on n that

$$\langle t_0| \cdots |t_n|t_{n+1}| \sim \langle u_0| \cdots |u_n|(q_n \otimes \mathrm{id}) \fatsemi t_{n+1}| \tag{7}$$

by only sliding identities on the last component. For $n = 0$ this is vacuously true because $t_0 = (\mathrm{id}_I \otimes \mathrm{id}) \fatsemi t_0$. Suppose that Equation (7) is true for n and we show that it holds for $n+1$ using the definition of q_{n+1} as quasi-total conditional, the range $r'_{n+1} \fatsemi \iota_{n+1}$ of c_{n+1} and that $\iota_{n+1} \fatsemi q_{n+1}$ is total by Lemma D.4.

$$\begin{aligned} &\langle t_0| \cdots |t_n|t_{n+1}|t_{n+2}| \\ &\sim \langle u_0| \cdots |u_n|(q_n \otimes \mathrm{id}) \fatsemi t_{n+1}|t_{n+2}| \\ &\sim \langle u_0| \cdots |u_n|c_{n+1} \triangleleft q_{n+1}|t_{n+2}| \\ &= \left\langle u_0| \cdots |u_n|c_{n+1} \triangleleft (r'_{n+1} \fatsemi \iota_{n+1} \fatsemi q_{n+1})|t_{n+2}\right| \\ &\sim \left\langle u_0| \cdots |u_n|c_{n+1} \triangleleft r'_{n+1}|((\iota_{n+1} \fatsemi q_{n+1}) \otimes \mathrm{id}) \fatsemi t_{n+2}\right| \\ &\sim \left\langle u_0| \cdots |u_n|c_{n+1} \triangleleft (r'_{n+1} \fatsemi \iota_{n+1})|(q_{n+1} \otimes \mathrm{id}) \fatsemi t_{n+2}\right| \end{aligned}$$

We have shown that Equation (7) holds for every $n \in \mathbb{N}$, which implies that, for every $n \in \mathbb{N}$

$$\langle t_0| \cdots |t_n| \sim \langle u_0 \mid \cdots |u_n| = \mathrm{obs}(\mathrm{proc}(\langle t_n \mid n \in \mathbb{N}\rangle)) \,.$$

□

Proof of Theorem 4.6. Proposition 4.3 constructs a monoidal functor $\mathrm{obs} \colon \mathrm{Causal}(\mathbf{C}) \to \mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}$ and Proposition 4.4 constructs, for every objects X and Y, a function $\mathrm{proc} \colon \mathrm{OSeq}_{(\mathrm{Tot}(\mathbf{C}),\mathbf{C})}(X,Y) \to \mathrm{Causal}(\mathbf{C})(X,Y)$. Lemmas E.7 and E.9 show that this functor is fully faithful with inverses given by proc. Identity-on-objects fully faithful functors are isomorphisms. □

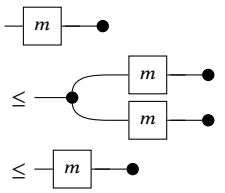
E.1 Proofs about conditionals and ranges

Lemma E.10. *Any cartesian category has quasi-total conditionals.*

Proof. A morphism $f \colon X \to A \times B$ can be split, using its determinism, as $f = \multimap \fatsemi ((f \fatsemi \pi_A) \times (f \fatsemi \pi_B))$. Then, $f \fatsemi \pi_B$ is a quasi-total conditional of f because it is total. Using this fact, the causality condition simplifies to that in Figure 5 and reduces causal processes to families of morphisms $A_0 \times \cdots \times A_n \to B_n$. □

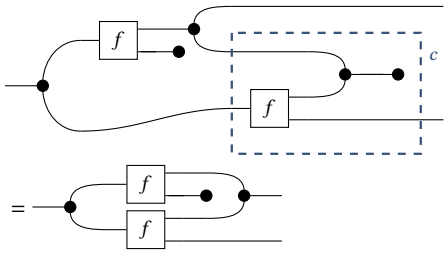
Lemma E.11 (From Lemma 4.10)**.** *The monoidal category of relations* **Rel** *has quasi-total conditionals (Figure 4).*

Proof. We use the syntax of cartesian bicategories of relations, which is sound and complete for **Rel** [CKWW08]. Every morphism in **Rel** is quasi-total.

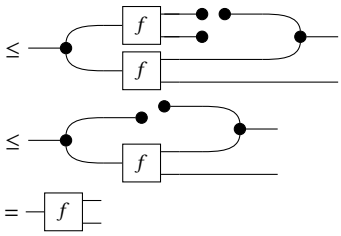


(8) (9)

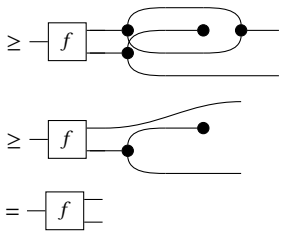
Equation (8) uses lax naturality of the copy morphism, while Equation (9) uses lax naturality of the discard morphism. The equation for conditionals can be simplified using the Frobenius equation.



We bound this morphism from above with f using adjointness of the discard and the codiscard, and lax naturality of the discard morphism.



We bound the same morphism also from below with f using lax naturality of the copy morphism, and adjointness of the copy with the cocopy.



□

Lemma E.12 (From Lemma 4.10)**.** *The monoidal category of relations* **Rel** *has ranges. For a relation $m \colon X \to A$, its range is given by*

$r' \colon X \times A \to R$ and $\iota \colon R \to X \times A$, with $R \coloneqq \{(x,a) \in X \times A : a \in m(x)\}$, defined below.

$$r'(x,a) \coloneqq \begin{cases} \{(x,a)\} & \text{if } a \in m(x) \\ \emptyset & \text{otherwise} \end{cases} \qquad \iota(x,a) \coloneqq \{(x,a)\}$$

Proof. The relation r' is deterministic and the relation ι is deterministic and total by definition. We check the first condition for ranges.

$$\begin{aligned} & m \triangleleft r(x) \\ &= \{(x',a',a) : a \in m(x) \wedge (x',a') \in r(x,a)\} \\ &= \{(x,a,a) : a \in m(x)\} \\ &= m \triangleleft \mathrm{id}(x) \end{aligned}$$

Similarly, we check the last condition. Suppose that $m \triangleleft c = m \triangleleft d$. Then, for all $x \in X$ and $y \in Y$,

$$\begin{aligned} & \{(b,a) \in A \times B : a \in m(x) \wedge b \in c(y,x,a)\} \\ &= \{(b,a) \in A \times B : a \in m(x) \wedge b \in d(y,x,a)\}\,, \end{aligned}$$

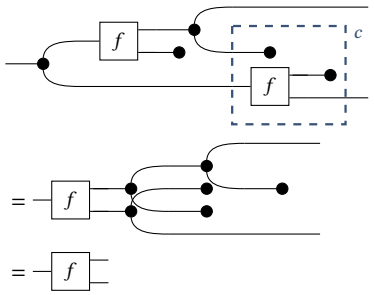
which implies that, for all $x \in X$,

$$\begin{aligned} & \{(x,a,b) : (a,x) \in R \wedge b \in c(y,x,a)\} \\ &= \{(x,a,b) : (a,x) \in R \wedge b \in c(y,x,a)\}\,. \end{aligned}$$

This corresponds to saying that $(\mathrm{id} \times \iota) \fatsemi c = (\mathrm{id} \times \iota) \fatsemi d$. □

Lemma E.13. *The monoidal category* **Par** *of partial functions has quasi-total conditionals (Figure 5).*

Proof. All morphisms in **Par** are deterministic. We use this to show that there are conditionals.



These conditionals are quasi-total because every morphism is deterministic and, in particular, $c \fatsemi \multimap$ also is so. □

Lemma E.14 (From Lemma 4.12)**.** *The monoidal category of partial functions* **Par** *has ranges. For a partial function $m \colon X \to A$, its range is given by $r' \colon X \times A \to R$ and $\iota \colon R \to X \times A$, with $R \coloneqq \{(x,a) \in X \times A : a = m(x)\}$, defined below.*

$$r'(x,a) \coloneqq \begin{cases} (x,a) & \text{if } a = m(x) \\ \bot & \text{otherwise} \end{cases} \qquad \iota(x,a) \coloneqq (x,a)$$

Proof. The ranges are defined as in **Rel**. Since **Par** is a subcategory of **Rel**, these ranges also satisfy the same properties. □

Lemma E.15 (From Lemma 4.14)**.** *The monoidal category* **ParStoch** *has quasi-total conditionals and ranges.*

Proof. The existence of quasi-total conditionals was shown by Di Lavore and Román [DLR23]. For a morphism $f \colon X \to A \otimes B$ in **ParStoch** a quasi-total conditional $c \colon X \otimes A \to B$ of f is defined as

$$c(b \mid x,a) = \frac{f(a,b \mid x)}{\sum_{b' \in B} f(a,b' \mid x)}; \quad c(\bot \mid x,a) = 0$$

whenever defined, and by $c(b \mid x,a) = 0$ and $c(\bot \mid x,a) = 1$ otherwise. See the next Lemma E.16, for the properties of ranges. □

Lemma E.16 (From Lemma 4.14)**.** *The monoidal category of partial stochastic functions* $\mathsf{kl}(\mathrm{D}_{\leq 1})$ *has ranges. For a partial stochastic function $m \colon X \to A$, its range is given by the deterministic morphisms $r' \colon X \times A \to R$ and $\iota \colon R \to X \times A$, with $R \coloneqq \{(x,a) \in X \times A : m(a \mid x) > 0\}$, defined as*

$$r'(x,a) \coloneqq \begin{cases} (x,a) & \text{if } m(a \mid x) > 0 \\ \bot & \text{otherwise} \end{cases} \qquad \iota(x,a) \coloneqq (x,a)$$

Proof. The morphism r' is deterministic and the morphism ι is deterministic and total by definition. We check the first condition for ranges in the case in which the computation succeeds and the one in which it fails.

$$\begin{aligned} & m \triangleleft r(x',a',a \mid x) \\ &= m(a \mid x) \cdot r(x',a' \mid x,a) \\ &= m \triangleleft \mathrm{id}(x',a',a \mid x) \\ & m \triangleleft r(\bot \mid x) \\ &= m(\bot \mid x) + \sum_{a \in A} m(a \mid x) \cdot r(\bot \mid x,a) \\ &= m \triangleleft \mathrm{id}(\bot \mid x) \end{aligned}$$

Similarly, we check the last condition. Suppose that $m \triangleleft c = m \triangleleft d$. Then, for all $x \in X$, $y \in Y$, $a \in A$ and $b \in B$,

$$m(a \mid x) \cdot c(b \mid y,x,a) = m(a \mid x) \cdot d(b \mid y,x,a)\,,$$

which implies that, if $m(a \mid x) > 0$, then $c(b \mid y,x,a) = d(b \mid y,x,a)$. This means that, if $(x,a) \in R$, then $c(b \mid y,x,a) = d(b \mid y,x,a)$. We can conclude that $(\mathrm{id} \times \iota) \fatsemi c = (\mathrm{id} \times \iota) \fatsemi d$. □

F EFFECTFUL MEALY MACHINES

Proof sketch of Proposition 5.3. Composition and identities are defined as in the usual state construction [KSW02]. Thus, they are associative and unital, making $\mathbf{Mealy}(\mathbf{V},\mathbf{P},\mathbf{C})$ a category. Whiskering is also defined as in the monoidal case and it is easy to check that it gives the monoidal actions required for the effectful structure. □

Proof sketch of Proposition 5.6. The trace coincides with the functor given by the universal property of $\mathbf{Mealy}(\mathbf{V},\mathbf{P},\mathbf{C})$. This is the free category with delayed feedback [DLGR+23, Section 5.2] and $\mathrm{Stream}(\mathbf{P},\mathbf{C})$ have delayed feedback by Proposition 3.7. In the monoidal case, the trace preserves the monoidal structure. For the same reasons, in the effectful case, it preserves whiskering. Then, the trace is an effectful functor. □

Proof sketch of Proposition 5.9. Two morphisms of effectful Mealy machines $\alpha \colon (U,i,f) \Rightarrow (V,j,g)$ and $\beta \colon (V,j,g) \Rightarrow (W,k,h)$ can be composed sequentially to obtain $\alpha \fatsemi \beta \colon (U,i,f) \Rightarrow$

(W, k, h). This composition clearly satisfies the equations in Figure 10. Two morphisms, $\alpha\colon (U_0, i_0, f_0) \Rightarrow (V_0, j_0, g_0)$ and $\beta\colon (U_1, i_1, f_1) \Rightarrow (V_1, j_1, g_1)$, can be composed in parallel into a morphism, $(\alpha \otimes \beta)\colon (U_0, i_0, f_0) \fatsemi (U_1, i_1, f_1) \Rightarrow (V_0, j_0, g_0) \fatsemi (V_1, j_1, g_1)$. We can check that tensoring forms a homomorphism of transition systems.

$$\left.\begin{array}{l} f_0(\mathtt{s}_0, \mathtt{x}) \rightsquigarrow \mathtt{s}_1, \mathtt{y} \\ f_1(\mathtt{u}_0, \mathtt{y}) \rightsquigarrow \mathtt{u}_1, \mathtt{z} \\ \mathtt{let}\ \mathtt{t}_1 = \alpha(\mathtt{s}_1) \\ \mathtt{let}\ \mathtt{v}_1 = \beta(\mathtt{u}_1) \end{array}\right\} = \left\{\begin{array}{l} \mathtt{let}\ \mathtt{t}_0 = \alpha(\mathtt{s}_0) \\ \mathtt{let}\ \mathtt{v}_0 = \beta(\mathtt{u}_0) \\ g_0(\mathtt{t}_0, \mathtt{x}) \rightsquigarrow \mathtt{t}_1, \mathtt{y} \\ g_1(\mathtt{v}_0, \mathtt{y}) \rightsquigarrow \mathtt{v}_1, \mathtt{z} \end{array}\right.$$

□

Proof of Theorem 5.10. We prove that the existence of a morphism $\alpha\colon (U, i, f) \to (V, j, g)$ implies the equality $\alpha \cdot [\![g]\!] = [\![f]\!]$. We proceed by coinduction, noting that

$$(\alpha \cdot [\![g]\!])^\circ = (\alpha \otimes \mathrm{id}) \fatsemi g = f \fatsemi (\alpha \otimes \mathrm{id}) = [\![f]\!]^\circ \fatsemi (\alpha \otimes \mathrm{id});$$

and that then, by coinductive hypothesis, $\alpha \cdot (\alpha \cdot [\![g]\!])^+ = \alpha \cdot [\![g]\!] = [\![f]\!] = [\![f]\!]^+$. In particular, this implies that $\mathrm{tr}(U, i, f) = \mathrm{tr}(V, j, g)$ whenever $j = i \fatsemi \alpha$.

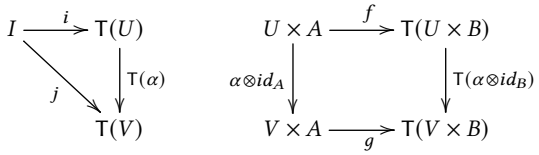
We have shown that the existence of a morphism between two Mealy machines implies trace equivalence; we conclude that the existence of a zig-zag of morphisms also implies trace equivalence, by transitivity of equality in $\mathsf{Stream}(\mathbf{P}, \mathbf{C})$. □

F.1 T-Mealy machines

Definition F.1. Let $\mathsf{T}\colon \mathbf{Set} \to \mathbf{Set}$ be a commutative monad and A, B two sets. A T-*Mealy machine* with inputs in A and outputs in B is a triple (U, i, f) consisting of a set U, a function $i\colon I \to \mathsf{T}(U)$ and a function $f\colon U \times A \to \mathsf{T}(U \times B)$.

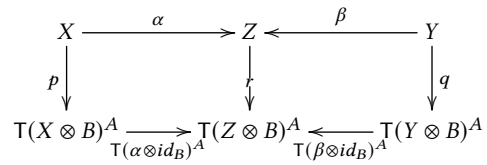
The set U represents the state space, i the initial state and f the effectful transition function. We will often write $(U, i, f)\colon A \to B$ to mean that inputs are in A and outputs in B.

Definition F.2. Let (U, i, f) and (V, j, g) be two T-Mealy machines with inputs in A and outputs in B. A *morphism* of Mealy machines $\alpha\colon (U, i, f) \Rightarrow (V, j, g)$ is a function $\alpha\colon U \to V$ such that the following two diagrams commute.



The machines (U, i, f) and (V, j, g) are *isomorphic*, written $(U, i, f) \cong (V, j, g)$ if there exists an isomorphism $\alpha\colon (U, i, f) \Rightarrow (V, j, g)$.

Proof of Proposition 5.17. First recall that, since T preserves weak pullbacks, then $i'(\bullet)$ and $j'(\bullet)$ are bisimilar in the sense of [Rut00] if and only if there exists a cospan of coalgebras morphism



such that $\alpha(i'(\bullet)) = \beta(j'(\bullet))$. If this is the case, one takes k to be $\eta_Z(\alpha(i'(\bullet)))$ and concludes by observing that $(U, i, f) \overset{\alpha}{\Rightarrow} (W, k, h) \overset{\beta}{\Leftarrow} (V, j, g)$ is a cospan of morphisms of Mealy machines.

For the other direction, assume to have a cospan $(U, i, f) \overset{\alpha}{\Rightarrow} (W, k, h) \overset{\beta}{\Leftarrow} (V, j, g)$. Since $i' \fatsemi \eta_X$ then, by naturality of η, also k should factor as $k' \fatsemi \eta_Z$. In particular k' should be equal to both $i' \fatsemi \alpha$ and $j' \fatsemi \beta$. Thus one has a cospan of coalgebra morphism as above. □