A CANONICAL ALGEBRA OF OPEN TRANSITION SYSTEMS (EXTENDED ABSTRACT)

ELENA DI LAVORE, ALESSANDRO GIANOLA, MARIO ROMÁN, NICOLETTA SABADINI, AND PAWEŁ SOBOCIŃSKI

ABSTRACT. Feedback and state are closely interrelated concepts. Categories with feedback, originally proposed by Katis, Sabadini and Walters, are a weakening of the notion of traced monoidal categories, with several pertinent applications in computer science. The construction of the *free* such categories has appeared in several different contexts, and can be considered as *state bootstrapping*. We show that a categorical algebra for *open transition systems*, **Span**(**Graph**)$_*$, also due to Katis, Sabadini and Walters, is the free category with feedback over **Span**(**Set**). Intuitively, this algebra of transition systems is obtained by adding state to an algebra of predicates, and therefore **Span**(**Graph**)$_*$ is, in this sense, the canonical such algebra.

1. INTRODUCTION

1.1. **State from feedback.** A remarkable fact from electronic circuit design is how data-storing components can be built out of a combination of *stateless components* and *feedback*. A famous example is the (set-reset) "NOR latch": a circuit with two stable configurations that stores one bit of information.

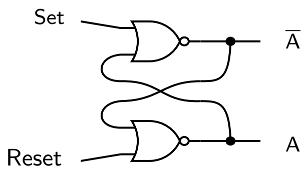


FIGURE 1. NOR latch.

The NOR latch is controlled by two inputs, Set and Reset. Activating the first sets the output value to $\mathsf{A} = 1$; activating the second makes the output value return to $\mathsf{A} = 0$. This change is permanent: even when both Set and Reset are deactivated, the feedback loop maintains the last value the circuit was set to[1]—to wit, a bit of data has been conjured out of thin air. In this paper we show that this can be seen as an instance of a more abstract phenomenon: the universal way of adding feedback to a theory of processes consists of endowing each process with a *state space*.

Indeed, there is a natural weakening of the notion of traced monoidal categories called *categories with feedback* [30]. The construction of the *free* category with feedback coincides with a "state-bootstrapping" construction, $\mathsf{St}(\bullet)$, that appears in several different contexts in the literature [7, 23, 26]. We recall this construction and its mathematical status (Theorem 2.5), which can be summed up by the following intuition:

$$\text{Theory of Processes} + \text{Feedback} = \text{Theory of Stateful Processes}.$$

1.2. **The algebra of transition systems.** Our primary focus is the **Span**(**Graph**) model of concurrency, introduced in [27] as a categorical algebra of *communicating*

[1]In its original description: *"the relay is designed to produce a large and* ***permanent*** *change in the current flowing in an electrical circuit by means of a small electrical stimulus received from the outside"* ([12], emphasis added).

state machines, or — equivalently — *open transition systems*. Open transition systems do not interact by input-output message passing, but by synchronization, producing a *simultaneous change of state*. This corresponds to a composition of spans, realized by taking a pullback in **Graph**. The dual algebra of **Cospan**(**Graph**) was introduced in [29]. It complements **Span**(**Graph**) by adding the operation of *communicating-sequential composition* [17].

Informally, a component of **Span**(**Graph**) is a state machine with states and transitions, i.e. a finite graph given by the 'head' of the span. The transition system is equipped with interfaces or *communication ports*, and every transition is labeled by the effect it produces in *all* its interfaces. We give examples below.

1.3. **Stateful and stateless components.** In Figure 2, we depict two open transition systems as arrows of **Span**(**Graph**). The first represents a NOR gate $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$. The diagram below left (Figure 2) is a graphical rendering of the corresponding span $\mathbb{B} \times \mathbb{B} \leftarrow N \to \mathbb{B}$, where $\mathbb{B}$ is considered as a single-vertex graph with two edges, corresponding to the signals $\{0, 1\}$, the *unlabeled* graph depicted within the bubble is N, and the labels witness the action of two homomorphisms, respectively $N \to \mathbb{B} \times \mathbb{B}$ and $N \to \mathbb{B}$. Here each transition represents one of the valid input/output configurations of the gate. NOR gates are *stateless* components, since their transition graph N has a single vertex.

The second component is a span $L = \{\mathsf{Set}, \mathsf{Reset}, \mathsf{Idle}\} \to \{\mathsf{A}, \overline{\mathsf{A}}\} = R$ that models a set-reset latch. The diagram below right (Figure 2), again, is a convenient way of denoting the relevant span $L \leftarrow D \to R$. Latches store one bit of information, they are *stateful components*; consequently, their transition graph has two states.

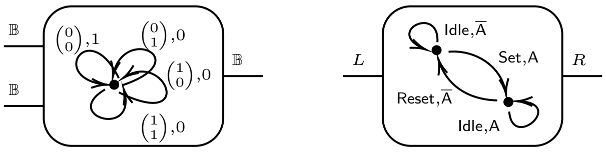


FIGURE 2. A NOR gate and set-reset latch, in **Span**(**Graph**).

In both cases, the boundaries on **Span**/**Cospan**(**Graph**) are stateless: indeed, they are determined by a mere set – the self-loops of a single-vertex graph. This is a restriction that occurs rather frequently: the important subcategory of **Span**(**Graph**), the one that we can clearly conceptually explain as *transition systems with interfaces*, is the full subcategory of **Span**(**Graph**) restricted to objects that are single-vertex graphs, which we denote by **Span**(**Graph**)$_*$. Analogously, the relevant subcategory of **Cospan**(**Graph**) is **Cospan**(**Graph**)$_*$, the full subcategory on sets, or graphs with an empty set of edges.

Definition. **Span**(**Graph**)$_*$ is the full subcategory of **Span**(**Graph**) with objects the single-vertex graphs.

The problem with **Span**(**Graph**)$_*$ is that it is somewhat mysterious from the categorical point of view; the morphisms are graphs, but the boundaries are given by sets. *Decorated* and *structured* spans and cospans [14, 3] were introduced as theoretical frameworks to capture such phenomena, which occur frequently when composing network structures. Nevertheless, they do not quite answer the question of *why* such examples do arise naturally.

1.4. **Canonicity and our original contribution.** Universal constructions, such as "state-bootstrapping" $\mathsf{St}(\bullet)$, characterize the object of interest up to equivalence, making it *the canonical object* satisfying some properties. This is the key to avoiding the problem outlined by Abramsky [1]: because of the lack of consensus about the intrinsic primitives of concurrency, we risk making our results too dependent on a specific syntax. It is thus important to characterize existing modeling formalisms for concurrent systems in terms of universal properties.

The main contribution of this paper is the characterization of $\mathbf{Span}(\mathbf{Graph})_*$ in terms of a universal property: it is equivalent to the free category with feedback over the category of spans of functions. We now state this more formally:

Theorem. The free category with feedback over $\mathbf{Span}(\mathbf{Set})$ is isomorphic to the category $\mathbf{Span}(\mathbf{Graph})_*$, the full subcategory of $\mathbf{Span}(\mathbf{Graph})$ given by single-vertex graphs. That is, there is an isomorphism of categories $\mathsf{St}(\mathbf{Span}(\mathbf{Set})) \cong \mathbf{Span}(\mathbf{Graph})_*$.

Given that $\mathbf{Span}(\mathbf{Set})$, the category of spans of functions, can be considered an *algebra of predicates* [4, 10], the high level intuition that summarizes our main contribution (Theorem 3.8) can be stated as:

$$\text{Algebra of Predicates} + \text{Feedback} = \text{Algebra of Transition Systems.}$$

We similarly prove (Section 3.4) that the free category with feedback over the category of cospans of functions, $\mathbf{Cospan}(\mathbf{Set})$, is equivalent to $\mathbf{Cospan}(\mathbf{Graph})_*$, the full subcategory on discrete graphs of $\mathbf{Cospan}(\mathbf{Graph})$.

1.5. **Related Work.** $\mathbf{Span}/\mathbf{Cospan}(\mathbf{Graph})$ has been extensively used for the modeling of concurrent systems [27, 29, 40, 41, 9, 37, 17, 15, 16]. Similar approaches to compositional modeling of networks have used *decorated* and *structured cospans* [14, 3]. Despite this, $\mathbf{Span}(\mathbf{Graph})_*$ has not previously been characterized in terms of a universal property.

In [30], the $\mathsf{St}(\bullet)$ construction (under a different name) is exhibited as the free *category with feedback*. Categories with feedback have been arguably under-appreciated but, at the same time, the $\mathsf{St}(\bullet)$ construction has made multiple appearances as a "state bootstrapping" technique across the literature. The $\mathsf{St}(\bullet)$ construction is used to describe a string diagrammatic syntax for *concurrency theory* in [7]; a variant of it had been previously applied in the setting of *cartesian bicategories* in [26]; and it was again rediscovered to describe a *memoryful geometry of interaction* in [23]. However, a coherent account of both categories with feedback and their relation with these stateful extensions has not previously appeared. This motivates our extensive preliminaries in Sections 2.1 and 2.2.

1.6. **Synopsis.** Section 2 contains preliminary discussions on traced monoidal categories and categories with feedback; it explicitly describes $\mathsf{St}(\bullet)$, the free category with feedback. It collects mainly expository material. Section 3 exhibits a universal property for the $\mathbf{Span}(\mathbf{Graph})_*$ and $\mathbf{Cospan}(\mathbf{Graph})_*$ models of concurrency and Section 3.5 discusses a specific application.

1.7. **Conventions.** We write composition of morphisms in diagrammatic order, $(f;g)$. When describing morphisms in a symmetric monoidal category whose input and output are known, we omit the associators and unitors, implicitly using the coherence theorem for monoidal categories.

2. Preliminaries: categories with feedback

Categories with feedback were introduced in [30], and motivated by examples such as *Elgot automata* [13], *iteration theories* [6] and *continuous dynamical systems* [28]. We recall the details below, contrast them with the stronger notion of *traced monoidal categories* in Section 2.2, discuss the relationship between feedback and delay in Section 2.3, recall the construction of a free category with feedback in Section 2.4, and give examples in Section 2.5.

2.1. Categories with feedback.

A *feedback operator*, $\mathsf{fbk}(\bullet)$, takes a morphism $S \otimes A \to S \otimes B$ and *"feeds back"* one of its outputs to one of its inputs of the same type, yielding a morphism $A \to B$ (Figure 3, left). When using string diagrams, we depict the action of the feedback operator as a loop with a double arrowtip (Figure 3, right).

$$\frac{f \colon S \otimes A \to S \otimes B}{\mathsf{fbk}_S(f) \colon A \to B}$$

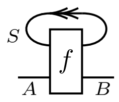


Figure 3. Type and graphical notation for the operator $\mathsf{fbk}_S(\bullet)$.

Capturing a reasonable notion of feedback requires the operator to interact nicely with the flow imposed by the structure of a symmetric monoidal category. This interaction is expressed by a few straightforward axioms.

Definition 2.1. A *category with feedback* [30] is a symmetric monoidal category $\mathbf{C}$ endowed with an operator $\mathsf{fbk}_S \colon \mathbf{C}(S \otimes A, S \otimes B) \to \mathbf{C}(A, B)$, which satisfies the following axioms (A1-A5, see also Figure 4).

(A1). *Tightening*, $u; \mathsf{fbk}_S(f); v = \mathsf{fbk}_S((\mathrm{id} \otimes u); f; (\mathrm{id} \otimes v))$.
(A2). *Vanishing*, $\mathsf{fbk}_I(f) = f$.
(A3). *Joining*, $\mathsf{fbk}_T(\mathsf{fbk}_S(f)) = \mathsf{fbk}_{S \otimes T}(f)$.
(A4). *Strength*, $\mathsf{fbk}_S(f) \otimes g = \mathsf{fbk}_S(f \otimes g)$.
(A5). *Sliding*, $\mathsf{fbk}_T(f; (h \otimes \mathrm{id})) = \mathsf{fbk}_S((h \otimes \mathrm{id}); f)$, for $h \colon S \to T$ any isomorphism.

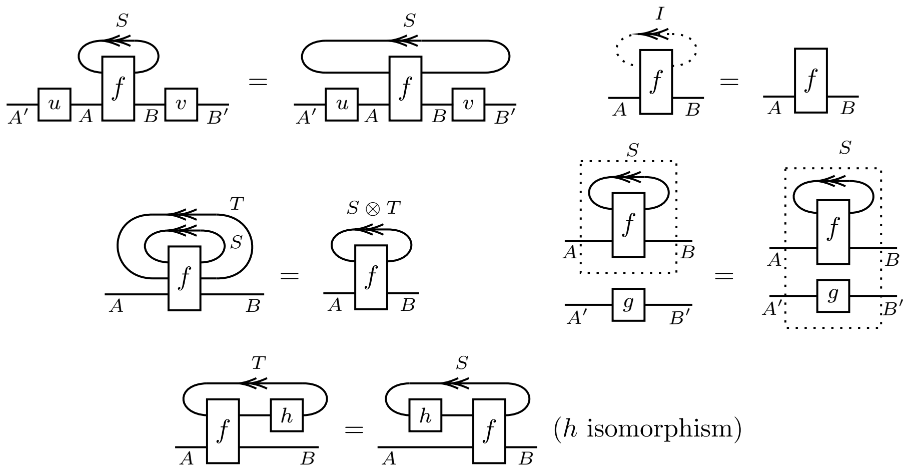


Figure 4. Diagrammatic depiction of the axioms of feedback.

The natural notion of homomorphism between categories with feedback is that of a symmetric monoidal functor that moreover preserves the feedback structure. These are called *feedback functors*.

Definition 2.2. A *feedback functor* $F\colon \mathbf{C} \to \mathbf{D}$ between two categories with feedback $(\mathbf{C}, \mathsf{fbk}^{\mathbf{C}})$ and $(\mathbf{D}, \mathsf{fbk}^{\mathbf{D}})$ is a strong symmetric monoidal functor such that

$$F(\mathsf{fbk}^{\mathbf{C}}_{S}(f)) = \mathsf{fbk}^{\mathbf{D}}_{F(S)}(\mu; Ff; \mu^{-1}),$$

where $\mu_{A,B}\colon F(A) \otimes F(B) \to F(A \otimes B)$ is the structure morphism of the strong monoidal functor F. We denote by $\mathsf{Feedback}$ the category of (small) categories with feedback and feedback functors between them. There exists a forgetful functor $\mathcal{U}\colon \mathsf{Feedback} \to \mathsf{SymMon}$.

2.2. **Traced monoidal categories.** Categories with feedback are a weakening of the well known traced monoidal categories. Between them, there is an intermediate notion called *right traced category* [38] that strengthens the sliding axiom from isomorphisms to arbitrary morphisms. This first extension would be already too strong for our purposes later in Section 2.4: we would be unable to define a *state space* up to isomorphism. However, the more conceptual difference of traced monoidal categories is the "yanking axiom" (in Figure 5). Indeed, strengthening the sliding axiom and adding the yanking axiom yields the definition of traced monoidal category.

Traced monoidal categories are widely used in theoretical computer science. Since their conception [24] as an abstraction of the *trace* of a matrix in linear algebra, they have been used in linear logic and geometry of interaction [1, 18, 19], programming language semantics [21], semantics of recursion [2] and fixed point operators [22, 5].

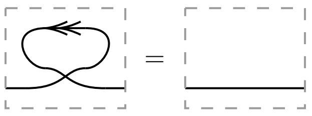


FIGURE 5. The yanking axiom.

Traces are thus undeniably important, but it is questionable whether we really want to always impose *all* of their axioms. Specifically, we will be concerned with the *yanking axiom* that states that $\mathsf{tr}(\sigma) = \mathrm{id}$. The yanking axiom is incontestably elegant from the geometrical point of view: strings are "pulled", and feedback (depicted as a loop with an arrowtip) disappears (Figure 5). However, if feedback can disappear without leaving any imprint, that must mean that it is *instantaneous*: its output necessarily mirrors its input.[2] Importantly for our purposes, this seems to imply that a feedback satisfying the yanking equation is "memoryless", or "stateless".

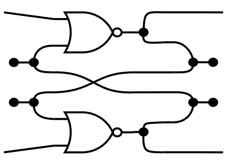


FIGURE 6. Diagram for the NOR latch, modeled with a trace in **Span**(**Graph**).

Consider again the NOR latch from Figure 1. We have seen how to model NOR gates in **Span**(**Graph**) in Figure 2, and the algebra of **Span**(**Graph**) does include a trace (see Figure 6, later detailed in Section 3.2). However, imitating the real-world behavior of the NOR latch with *just* a trace is unsatisfactory: the trace of **Span**(**Graph**) is built out of stateless components, and tracing stateless components yields again a stateless component.

In engineering and computer science, instantaneous feedback is actually a rare concept; a more common notion is that of *guarded feedback*. Consider *signal flow*

[2]In other words, traces are used to talk about processes in *equilibrium*, processes that have reached a *fixed point*. A theorem by Hasegawa [22] and Hyland [5] corroborates this interpretation: a trace in a cartesian category corresponds to a *fixpoint operator*.

graphs [39, 33]: their categorical interpretation in [8] models feedback not by the usual trace, but by a trace "guarded by a register", that *delays the signal* and violates the yanking axiom (see Remark 7.8 in *op.cit.*).

The component that trace misses in such examples is a *delay*.

2.3. **Delay and feedback.** The main difference between categories with feedback and traced monoidal categories is the failure of the yanking axiom. Consider the process that only "feeds back" the input to itself and then uses that "fed back" input to produce the output. This process, $\partial_A := \mathsf{fbk}_A(\sigma_{A,A})$, is called *delay endomorphism*. The *yanking axiom* of traced monoidal categories states that the delay is equal to the identity, $\mathsf{tr}_A(\sigma_{A,A}) = \mathrm{id}$, which is not necessarily true for *categories with feedback*. In that sense, a non-trivial delay is what sets apart categories with feedback from traced monoidal categories.

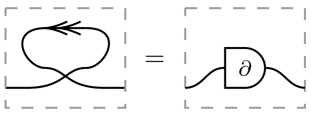


FIGURE 7. Definition of *delay*.

This interpretation of feedback as the combination of *trace* and *delay* can be made into a theorem when the category has enough structure. *Compact closed categories* are traced monoidal categories where every object A has a dual $A^\star$ and the trace is constructed from two pieces $\varepsilon\colon A \otimes A^\star \to I$ and $\eta\colon I \to A^\star \otimes A$. Even if not every traced monoidal category is compact closed, it is true that every traced monoidal category embeds fully faithfully into a compact closed category.[3] In a compact closed category, a feedback operator is necessarily a trace "guarded" by a *delay*.

Proposition 2.3 (Feedback from delay [7])**.** *Let* $\mathbf{C}$ *be a compact closed category with* $\mathsf{fbk}^{\mathbf{C}}$ *a feedback operator that takes a morphism* $S \otimes A \to S \otimes B$ *to a morphism* $A \to B$*, satisfying the axioms of feedback (in Figure 4) but possibly failing to satisfy the yanking axiom (Figure 5) of traced monoidal categories. Then the feedback operator is necessarily of the form*

$$\mathsf{fbk}^{\mathbf{C}}_S(f) := (\varepsilon \otimes \mathrm{id}); (\mathrm{id} \otimes f); (\mathrm{id} \otimes \partial_S \otimes \mathrm{id}); (\eta \otimes \mathrm{id})$$

where $\partial_A\colon A \to A$ *is a family of endomorphisms satisfying*

- $\partial_A \otimes \partial_B = \partial_{A\otimes B}$ *and* $\partial_I = \mathrm{id}$*, and*
- $\partial_A ; h = h ; \partial_B$ *for each isomorphism* $h\colon A \cong B$*.*

In fact, any family of morphisms ∂_A *satisfying these properties determines uniquely a feedback operator that has* ∂_A *as its delay endomorphisms.*

Consider one more time the NOR latch from Figure 1. The algebra of **Span**(**Graph**) does also include a feedback operator that is *not* a trace. This feedback operator is indeed canonical, in that it is the one that makes **Span**(**Graph**) the canonical category with feedback containing spans of functions. Imitating the real-world behavior of the NOR latch is finally possible: one of the components that builds up this feedback (and in fact, the only difference with the previous trace) is a *stateful* delay component. The emergence of state from feedback is witnessed by the $\mathsf{St}(\bullet)$ construction, which we recall below.

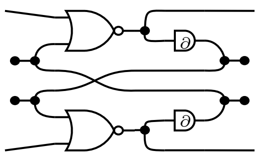


FIGURE 8. NOR latch with feedback.

[3]This is the **Int** construction from [24].

2.4. $\mathsf{St}(\bullet)$, the free category with feedback.

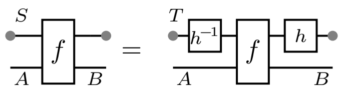


FIGURE 9. We depict stateful processes by marking the space state.

In this section, we identify the construction that yields the free category with feedback over a symmetric monoidal category. The $\mathsf{St}(\bullet)$ construction is a general way of endowing a system with state. It appears multiple times in the literature in slightly different forms: it constructs a stateful resource calculus in [7]; a variant is used for geometry of interaction in [23]; it coincides with the free category with feedback presented in [30]; and yet another, slightly different formulation was given in [26].

Definition 2.4 (Category of stateful processes, [30]). Let $(\mathbf{C}, \otimes, I)$ be a symmetric monoidal category. We write $\mathsf{St}(\mathbf{C})$ for the category having the same objects as $\mathbf{C}$ but where morphisms $A \to B$ are pairs $(S \mid f)$, consisting of a *state space* $S \in \mathbf{C}$ and a morphism $f \colon S \otimes A \to S \otimes B$. We consider morphisms up to isomorphism classes of their state space, and thus

$$(S \mid f) = (T \mid (h^{-1} \otimes \mathrm{id}); f; (h \otimes \mathrm{id})), \qquad \text{for any isomorphism } h \colon S \cong T.$$

When depicting a stateful process (Figure 9), we explicitly mark the strings forming the *state space*. That is, an equivalence class will be depicted as any of its representatives plus some strings marked.

We define the *identity stateful process* on $A \in \mathbf{C}$ as $(I \mid \mathrm{id}_{I \otimes A})$. *Sequential composition* of the two stateful processes $(S \mid f) \colon A \to B$ and $(T \mid g) \colon B \to C$ is defined by $(S \mid f); (T \mid g) = (S \otimes T \mid (\sigma \otimes \mathrm{id}); (\mathrm{id} \otimes f); (\sigma \otimes \mathrm{id}); (\mathrm{id} \otimes g))$. *Parallel composition* of the two stateful processes $(S \mid f) \colon A \to B$ and $(S' \mid f') \colon A' \to B'$ is defined by $(S \mid f) \otimes (S' \mid f') = (S \otimes S' \mid (\mathrm{id} \otimes \sigma \otimes \mathrm{id}); (f \otimes f'); (\mathrm{id} \otimes \sigma \otimes \mathrm{id}))$.

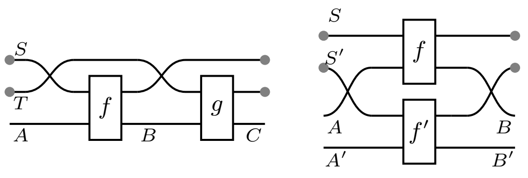


FIGURE 10. Sequential and parallel composition of stateful processes.

This defines a symmetric monoidal category. Moreover, it is a category with feedback with the operator $\mathsf{store}_T(S \mid f) := (S \otimes T \mid f)$.

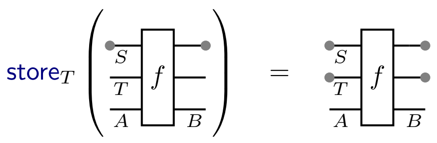


FIGURE 11. The $\mathsf{store}(\bullet)$ operation, in diagrammatic terms.

Theorem 2.5. [30] *The category $\mathsf{St}(\mathbf{C})$, endowed with the $\mathsf{store}(\bullet)$) operator, is the free category with feedback over a symmetric monoidal category $\mathbf{C}$.*

2.5. **Examples.** Our first source of examples is *traced monoidal categories*. The axioms of feedback are a strict weakening of the axioms of trace, and every traced category is automatically a category with feedback. A more interesting source of examples is the $\mathsf{St}(\bullet)$ construction we just defined.

Example 2.6. A *Mealy (or deterministic) transition system* with boundaries A and B, and state space S was defined [34, §2.1] to be just a function $f \colon S \times A \to S \times B$. Consider $\mathsf{St}(\mathbf{Set})$, the free category with feedback over the monoidal structure of sets with the cartesian product

We take *Mealy transition systems* to be morphisms of $\mathsf{St}(\mathbf{Set})$, which are functions of that signature up to isomorphism of the state space. Mealy transitions compose sequentially and in parallel following Definition 2.4, and they form a category with feedback $\mathbf{Mealy} \coloneqq \mathsf{St}(\mathbf{Set})$.

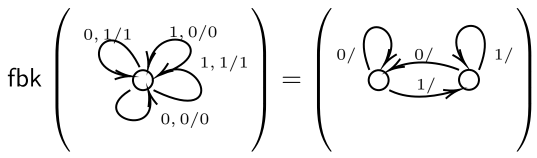


FIGURE 12. Feedback of a Mealy transition system. Every transition has a label i/o indicating inputs (i) and outputs (o).

The feedback operator of Mealy transitions *transforms* input/output pairs into states. Figure 12 is an example of what this means: an automaton with a single state becomes an automaton with two states and each transition $(s_i, i/s_o)$ becomes a transition $(i/)$ from s_i to s_o. The characterization $\mathbf{Span}(\mathbf{Graph})_* \cong \mathsf{St}(\mathbf{Span}(\mathbf{Set}))$ that we prove in Section 3 lifts the inclusion $\mathbf{Set} \to \mathbf{Span}(\mathbf{Set})$ to a feedback preserving functor $\mathbf{Mealy} \to \mathbf{Span}(\mathbf{Graph})_*$. This inclusion embeds a deterministic transition system into the graph that determines it. Indeed, it is traditional to depict automata as state/transition graphs.

Similarly, when we consider $\mathbf{Set}$ to be the monoidal structure of sets with the disjoint union, the notion we recover is that of an *Elgot automaton* [13], given by a transition function $S + A \to S + B$. These categories of transition systems motivate the work in [26, 30].

Example 2.7. A linear dynamical system with inputs in $\mathbb{R}^n$, outputs in $\mathbb{R}^m$ and state space in $\mathbb{R}^k$ is given by a matrix $\left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in \mathbf{Mat}_{\mathbb{R}}(k+m, k+n)$ [25]. Two linear dynamical systems $\left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} A' & B' \\ C' & D \end{smallmatrix}\right)$ are considered equal whenever there is an invertible matrix $H \in \mathbf{Mat}_{\mathbb{R}}(k,k)$ such that $A' = H^{-1}AH$, $B' = BH$, and $C' = H^{-1}C$.

Linear dynamical systems are morphisms of a category with feedback which coincides with $\mathsf{St}(\mathbf{Vect}_{\mathbb{R}}^{\oplus})$. The feedback operator is defined by

$$\mathsf{fbk}_l(k, \begin{pmatrix} A_1 & A_2 & B_1 \\ A_3 & A_4 & B_2 \\ C_1 & C_2 & D \end{pmatrix}) = (k + l, \begin{pmatrix} A_1 & A_2 & B_1 \\ A_3 & A_4 & B_2 \\ C_1 & C_2 & D \end{pmatrix}),$$

where $\begin{pmatrix} A_1 & A_2 & B_1 \\ A_3 & A_4 & B_2 \\ C_1 & C_2 & D \end{pmatrix} \in \mathbf{Mat}_{\mathbb{R}}(k + l + m, k + l + n)$.

3. Span(Graph): an algebra of transition systems

$\mathbf{Span}(\mathbf{Graph})$ [27] is an algebra of "open transition systems". It has applications in *concurrency theory* and *verification* [26, 27, 29, 31, 17], and has been recently applied to biological systems [15, 16]. Just as ordinary Petri nets have an underlying (firing) semantics in terms of transition systems, $\mathbf{Span}(\mathbf{Graph})$ is used as a semantic universe for a variant of open Petri nets, see [41, 9].

An *open transition system* is a morphism of $\mathbf{Span}(\mathbf{Graph})$: it consists of a graph endowed with two *boundaries* or *communication ports*; each transition of the graph has an effect on each boundary, and this data is used to synchronize a network of multiple transition systems. This conceptual picture actually describes a subcategory, $\mathbf{Span}(\mathbf{Graph})_*$, where boundaries are described by mere sets, accounting for the alphabets of signals that open transition systems synchronize on. In this section we recall the details of $\mathbf{Span}(\mathbf{Graph})_*$ and show that it is universal in the following sense:

$\mathbf{Span}(\mathbf{Graph})_*$ is the free category with feedback over $\mathbf{Span}(\mathbf{Set})$.

3.1. **The algebra of Span(Graph).**

Definition 3.1. A *span* [4, 10] from A to B, both objects of a category $\mathbf{C}$, is a pair of morphisms with a common domain, $A \leftarrow E \rightarrow B$. The object E is the "head" of the span, and the morphisms are the left and right "legs", respectively.

When the category $\mathbf{C}$ has pullbacks, we can sequentially compose two spans $A \leftarrow E \rightarrow B$ and $B \leftarrow F \rightarrow C$ into a span $A \leftarrow E \times_B F \rightarrow C$. Here, $E \times_B F$ is the pullback of E and F along B: for instance, in the category $\mathbf{Set}$ of sets and functions, $E \times_B F$ is the subset of $E \times F$ given by pairs whose two components have the same image on B.

Definition 3.2. Let $\mathbf{C}$ be a category with pullbacks. $\mathbf{Span}(\mathbf{C})$ is the category that has the same objects as $\mathbf{C}$ and isomorphism classes of spans between them as morphisms. That is, two spans are considered *equal* if there is an isomorphism between their heads that commutes with both legs. Dually, let $\mathbf{C}$ be a category with pushouts. $\mathbf{Cospan}(\mathbf{C})$ is the category $\mathbf{Span}(\mathbf{C}^{op})$.

$\mathbf{Span}(\mathbf{C})$ is a symmetric monoidal category when $\mathbf{C}$ has products. The parallel composition of $A \leftarrow E \rightarrow B$ and $A' \leftarrow E' \rightarrow B'$ is given by the componentwise product $A \times A' \leftarrow E \times E' \rightarrow B \times B'$. An example is again $\mathbf{Span}(\mathbf{Set})$.

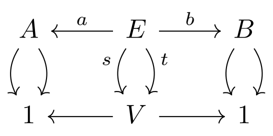
Definition 3.3. The category $\mathbf{Graph}$ has graphs $G = (s, t \colon E \rightrightarrows V)$ as objects. A morphism $G \rightarrow G'$ in this category is given by two functions $e \colon E \rightarrow E'$ and $v \colon V \rightarrow V'$ such that $e; s' = s; v$ and $e; t' = t; v$. In other words, it is the presheaf category on the diagram $(\bullet \rightrightarrows \bullet)$.

Recall, however, that we are not interested in the whole $\mathbf{Span}(\mathbf{Graph})$ but only in $\mathbf{Span}(\mathbf{Graph})_*$, the spans of graphs that have a graph $(A \rightrightarrows 1)$ with a single node on the boundaries.

Definition 3.4. An *open transition system*, a morphism of $\mathbf{Span}(\mathbf{Graph})_*$, is a span of sets $A \leftarrow E \rightarrow B$ where the head is the set of transitions of a graph $E \rightrightarrows V$ (see Figure 13). Two open transition systems are considered *equal* if there is an isomorphism between their graphs that commutes with the legs of the span.

Open transition systems whose graph $E \rightrightarrows V$ has a single vertex, $V = 1$, are called *"stateless"*.

Sequential composition (the *communicating-parallel operation* of [27]) of two open transition systems with spans $A \leftarrow E \rightarrow B$ and $B \leftarrow F \rightarrow C$ and graphs $E \rightrightarrows S$ and $F \rightrightarrows T$ yields the open transition system with span $A \leftarrow E \times_B F \rightarrow C$ and graph $E \times_B F \rightrightarrows S \times T$. This means that the only allowed transitions are those that synchronize E and F on the common boundary B.



$$\begin{tikzcd} A \arrow[d, bend right] \arrow[d, bend left] & E \arrow[l, "a"'] \arrow[r, "b"] \arrow[d, bend right, "s"'] \arrow[d, bend left, "t"] & B \arrow[d, bend right] \arrow[d, bend left] \\ 1 & V \arrow[l] \arrow[r] & 1 \end{tikzcd}$$

FIGURE 13. A morphism of $\mathbf{Span}(\mathbf{Graph})_*$.

Parallel composition (the *non communicating-parallel operation* of [27]) of two open transition systems with spans $A \leftarrow E \rightarrow B$ and $A' \leftarrow E' \rightarrow B'$ and graphs $E \rightrightarrows V$ and $E' \rightrightarrows V'$ yields the open transition system with span $A \times A' \leftarrow E \times E' \rightarrow B \times B'$ and graph $E \times E' \rightrightarrows V \times V'$.

3.2. **The components of Span(Graph).** Let us now detail some useful constants of the algebra of $\mathbf{Span}(\mathbf{Graph})$. We will illustrate how to use the algebra with an example in which we construct the NOR latch circuit from Figure 8.

Example 3.5. In this example, we model the circuit in Figure 8 in $\mathbf{Span}(\mathbf{Graph})_*$. The connectivity of the circuit is modeled with a Frobenius algebra [10] (, ,

$\bullet\!\!-$, $-\!\!\bullet$). The corresponding spans are constructed out of diagonals $A \to A \times A$ and units $A \to 1$.

$$(-\!\!\bullet\!\!<)_A = \{A \leftarrow A \to A \times A\} \qquad (-\!\!\bullet)_A = \{A \leftarrow A \to 1\}$$
$$(>\!\!\bullet\!\!-)_A = \{A \times A \leftarrow A \to A\} \qquad (\bullet\!\!-)_A = \{1 \leftarrow A \to A\}$$

These already induce a compact closed structure (and, therefore, a trace), given by the following spans.

$$(\bullet\!\!-\!\!\bullet\!\!<)_A = \{1 \leftarrow A \to A \times A\} \qquad (>\!\!\bullet\!\!-\!\!\bullet)_A = \{A \times A \leftarrow A \to 1\}$$

In general, any function $f\colon A \to B$ can be lifted covariantly to a span $A \leftarrow A \to B$ and contravariantly to a span $A \leftarrow B \to B$. Any span $A \leftarrow E \to B$ can be lifted to **Span**(**Graph**)$_*$ by making the head represent the graph $E \rightrightarrows 1$. We use this fact to obtain a NOR gate from the function `NOR`: $\mathbb{B} \times \mathbb{B} \to \mathbb{B}$ (Figure 2). However, components created like this have a single-vertex: they are *stateless*.

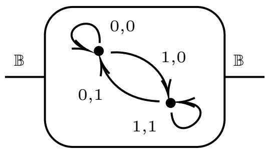


FIGURE 14. Delay morphism over the set $\mathbb{B} := \{0, 1\}$.

We will need a single stateful component to model our circuit, the delay

$$(-\!\boxed{\partial}\!-)_A = \left\{ \begin{tikzcd} & A \times A \arrow[dl, "\pi_2"'] \arrow[d, bend right, "\pi_1"'] \arrow[d, bend left, "\pi_2"] \arrow[dr, "\pi_1"] & \\ A & A & A \end{tikzcd} \right\}.$$

This is *not* an arbitrary choice. This is the canonical delay obtained from the feedback structure[4] in **Span**(**Graph**)$_*$ that gives its universal property.

The NOR latch circuit of Figure 8 is the composition of two NOR gates where the outputs of each gate have been copied and fed back as input to the other gate. The algebraic expression, in **Span**(**Graph**)$_*$, of this circuit is obtained by decomposing it into its components, as in Figure 15.

$$(\mathrm{id} \otimes \bullet\!\!-\!\!\bullet\!\!< \otimes \bullet\!\!-\!\!\bullet\!\!< \otimes\, \mathrm{id}); (\texttt{NOR} \otimes \sigma \otimes \texttt{NOR}); (-\!\!\bullet\!\!< \otimes\, \mathrm{id} \otimes -\!\!\bullet\!\!<)$$
$$; (\mathrm{id} \otimes \partial \otimes \mathrm{id} \otimes \partial \otimes \mathrm{id}); (\mathrm{id} \otimes >\!\!\bullet\!\!-\!\!\bullet \otimes >\!\!\bullet\!\!-\!\!\bullet \otimes\, \mathrm{id})$$

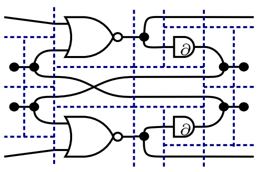


FIGURE 15. Decomposing the circuit.

The graph obtained by the computation of this expression, together with its transitions, is shown in Figure 16. This time, our model is indeed stateful. It has four states: two states representing a correctly stored signal, $\overline{\mathsf{A}} = (1, 0)$ and $\mathsf{A} = (0, 1)$; and two states representing transitory configurations $\mathsf{T}_1 = (0, 0)$ and $\mathsf{T}_2 = (1, 1)$.

[4]As in Proposition 2.3.

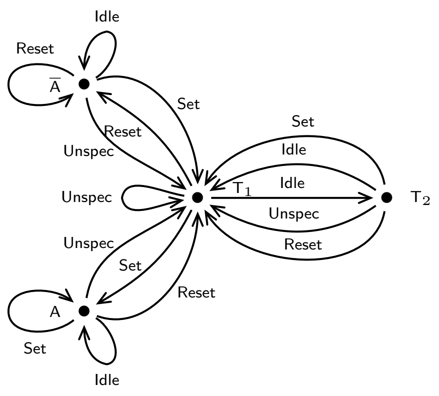


FIGURE 16. Span of graphs representing the NOR latch

We will be controlling the *left boundary*: it can receive a *set* signal, $\mathsf{Set} = \binom{1}{0}$; a *reset* signal, $\mathsf{Reset} = \binom{0}{1}$; none of the two, $\mathsf{Idle} = \binom{0}{0}$; or both of them at the same time, $\mathsf{Unspec} = \binom{1}{1}$, which is known to cause unspecified behavior in a NOR latch. The signal on the *right boundary*, on the other hand, is always equal to the state the transition goes to and does not provide any additional information. Knowing this, we omit it from the drawing in Figure 16.

In normal functioning, activating the signal Set makes the latch transition to the state A in two transition steps. Analogously, activating Reset makes the latch transition to $\overline{\mathsf{A}}$ again in two transition steps. After any of these two cases, deactivating all signals, Idle, keeps the last state.

Moreover, the (real-world) NOR latch has some unspecified behavior that gets also reflected in the graph: activating both Set and Reset at the same time, what we call Unspec, causes the circuit to enter an unstable state where it bounces between the states T_1 and T_2. Our modeling has reflected this "unspecified behavior" as expected.

3.2.1. *Feedback and trace.* In terms of feedback, the circuit of Figure 16 is equivalently obtained as the result of taking feedback over the following stateless morphism in Figure 17. We know that it is stateless because it is the composition of stateless morphisms.

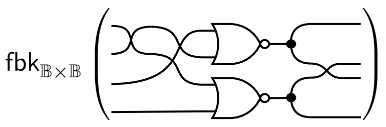


FIGURE 17. Applying $\mathsf{fbk}(\bullet)$ over the circuit gives the NOR latch.

But **Span(Graph)**$_*$ is also canonically traced: it is actually compact closed. What changes in the modeling if, over the same morphism, we would have used trace instead? As we argued back for Figure 6, we obtain a stateless transition system: it is given by a graph with a single vertex. The valid transitions can be now computed explicitly to be

$$\{(\mathsf{Unspec}, \mathsf{T}_1), (\mathsf{Idle}, \mathsf{A}), (\mathsf{Idle}, \overline{\mathsf{A}}), (\mathsf{Set}, \mathsf{A}), (\mathsf{Reset}, \overline{\mathsf{A}}).\}$$

These encode important information: they are the *equilibrium* states of the circuit. However, unlike the previous graph, this one would not get us the correct allowed transitions: under this modeling, our circuit could freely bounce between $(\mathsf{Idle}, \mathsf{A})$ and $(\mathsf{Idle}, \overline{\mathsf{A}})$, which is not the expected behavior of a NOR latch.

The fundamental piece making our modeling succeed the previous time was feedback derived from the stateful *delay.* The next section explains in which sense that feedback is canonical.

3.3. **Span(Graph) as a category with feedback.** This section presents our main theorem. We start by introducing the mapping that associates to each stateful span of sets the corresponding span of graphs. This mapping is well-defined and lifts to a functor $\mathsf{St}(\mathbf{Span}(\mathbf{Set})) \to \mathbf{Span}(\mathbf{Graph})$. Finally, we prove that it gives an isomorphism $\mathsf{St}(\mathbf{Span}(\mathbf{Set})) \cong \mathbf{Span}(\mathbf{Graph})_*$.

Proposition 3.6. *The composition of two stateful spans in the category* $\mathsf{St}(\mathbf{Span}(\mathbf{Set}))$,

$$S \times A \xleftarrow{\sigma,f} X \xrightarrow{\sigma',g} S \times B, \qquad T \times B \xleftarrow{\tau,h} Y \xrightarrow{\tau',h} T \times C$$

is given by the span $T \times S \times A \xleftarrow{\tau,\sigma,f} X \times_B Y \xrightarrow{\tau',\sigma',k} T \times S \times C$, *where* $X \times_B Y$ *is the pullback along* g *and* h.

Lemma 3.7. *The following assignment of* stateful processes *over* $\mathbf{Span}(\mathbf{Set})$ *to morphisms of* $\mathbf{Span}(\mathbf{Graph})$ *is well defined.*

$$K\left(S \;\middle|\; \begin{tikzcd} & E \arrow[dl, "{(s,a)}"'] \arrow[dr, "{(t,b)}"] & \\ S\times A & & S\times B \end{tikzcd} \right) := \left(\begin{tikzcd} A \arrow[d, bend right] \arrow[d, bend left] & E \arrow[l, "a"'] \arrow[r, "b"] \arrow[d, bend right, "s"'] \arrow[d, bend left, "t"] & B \arrow[d, bend right] \arrow[d, bend left] \\ 1 & S \arrow[l] \arrow[r] & 1 \end{tikzcd} \right)$$

Theorem 3.8. *There exists an isomorphism of categories*

$$\mathsf{St}(\mathbf{Span}(\mathbf{Set})) \cong \mathbf{Span}(\mathbf{Graph})_*.$$

That is, the free category with feedback over $\mathbf{Span}(\mathbf{Set})$ *is isomorphic to the full subcategory of* $\mathbf{Span}(\mathbf{Graph})$ *given by single-vertex graphs.*

3.4. **Cospan(Graph) as a category with feedback.** The previous results can be generalized to any category $\mathbf{C}$ with all finite limits. By taking $\mathbf{Graph}(\mathbf{C})$ to be the presheaf category of the diagram $(\bullet \rightrightarrows \bullet)$ in $\mathbf{C}$ and $\mathbf{Span}(\mathbf{Graph}(\mathbf{C}))_*$ the full subcategory on objects of the form $A \rightrightarrows 1$, we can prove the following result.

Theorem 3.9. *There exists an isomorphism of categories*

$$\mathsf{St}(\mathbf{Span}(\mathbf{C})) \cong \mathbf{Span}(\mathbf{Graph})_*.$$

That is, the free category with feedback over $\mathbf{Span}(\mathbf{C})$ *is equivalent to the full subcategory on* $\mathbf{Span}(\mathbf{Graph}(\mathbf{C}))$ *given by single-vertex graphs.*

$\mathbf{Cospan}(\mathbf{Graph})_*$ can be also characterized as a free category with feedback. We know that $\mathbf{Cospan}(\mathbf{Set}) \cong \mathbf{Span}(\mathbf{Set}^{op})$, we note that $\mathbf{Graph}(\mathbf{Set}^{op}) \cong \mathbf{Graph}$ (which has the effect of flipping edges and vertices), and we can use Theorem 3.9 because $\mathbf{Set}$ has all finite colimits.

Corollary 3.10. *There exists an isomorphism of categories* $\mathsf{St}(\mathbf{Cospan}(\mathbf{Set})) \cong \mathbf{Cospan}(\mathbf{Graph})_*$.

$\mathbf{Cospan}(\mathbf{Graph})$ is also compact closed and, in particular, traced. As in the case of $\mathbf{Span}(\mathbf{Graph})$, the feedback structure given by the universal property is different from the trace. In the case of $\mathbf{Cospan}(\mathbf{Graph})$, tracing has the effect of identifying the output and input vertices of the graph; while feedback adds an additional edge from the output to the input vertices.

3.5. **Syntactical presentation of Cospan(FinGraph).** The observation in Proposition 2.3 has an important consequence in the case of finite sets. We write $\mathbf{FinGraph}$ for $\mathbf{Graph}(\mathbf{FinSet})$. $\mathbf{Cospan}(\mathbf{FinSet})$ is the generic special commutative Frobenius algebra [32], meaning that any morphism written out of the operations of a special commutative Frobenius algebra and the structure of a symmetric monoidal category is precisely a cospan of finite sets (or, in other words, symmetric monoidal functors out of **Cospan**(**FinSet**) correspond to special commutative Frobenius algebras). But we also know that **Cospan**(**FinSet**), with an added generator to its PROP structure [7] (the delay, with the conditions given in Proposition 2.3), is St(**Cospan**(**FinSet**)), or, equivalently, **Cospan**(**FinGraph**). This means that any morphism written out of the operations of a special commutative Frobenius algebra plus a freely added generator of type $(-\boxed{\partial}-) \colon 1 \to 1$ is a morphism in **Cospan**(**FinGraph**)$_*$. This way, we recover one of the main results of [36] as a direct corollary of our characterization.

Proposition 3.11 (Proposition 3.2 of [36])**.** *The category* **Cospan**(**FinGraph**)$_*$ *is the generic special commutative Frobenius monoid with an added generator.*

Proof. It is known that the category **Cospan**(**FinSet**) is the generic special commutative Frobenius algebra [32]. Adding a free generator $(-\boxed{\partial}-) \colon 1 \to 1$ to its PROP structure corresponds to adding a family $(-\boxed{\partial}-)_n \colon n \to n$ with the conditions on Proposition 2.3. Now, Proposition 2.3 implies that adding such a generator to **Cospan**(**FinSet**) results in St(**Cospan**(**FinSet**)). Finally, we can use again Theorem 3.8 to conclude that St(**Cospan**(**FinSet**)) $\cong$ **Cospan**(**FinGraph**)$_*$. □

4. Conclusions and further work

We have characterized **Span**(**Graph**)$_*$, an algebra of open transition systems, as equivalent to the free category with feedback over the category of spans of functions. The $\mathsf{St}(\bullet)$ constuction is well-known as a technique of adding state to processes. In [30], it had been characterized as the free category with feedback under a different name. What was missing was a coherent and explicit connection between the two.

We have seen how the $\mathsf{St}(\bullet)$ construction creates categories of *transition systems* out of symmetric monoidal categories. We could also consider a generalization of this construction where, instead of quotienting by isomorphisms, we can quotient by arbitrary classes of morphisms selected by some strong monoidal functor. Our observation is that this generalized state construction can be rewritten compactly as a particular kind of colimit called a *coend*. In fact, let $F \colon \mathbf{D} \to \mathbf{C}$ be a strong monoidal functor, we can express the set of stateful morphisms quotiented by *sliding* in $\mathbf{D}$ as

$$\mathsf{St}_{\mathbf{D}}(\mathbf{C}) \coloneqq \int^{D \in \mathbf{D}} \hom(FD \otimes X, FD \otimes Y).$$

For instance, the original $\mathsf{St}(\bullet)$ construction is recovered from the inclusion functor of the subgroupoid of isomorphisms. The identity functor can be used to quotient processes by dinaturality. The forgetful **PointedSet** $\to$ **Set** can be used to construct automata with initial states.

All these possibilities deserve special attention, together with their potential applications, both to transition systems and stateful automata. For instance, obtaining stream transducers, **Stream**$(A) \to$ **Stream**(B), from transition systems, $S \times A \to S \times B$, is only possible with an initial state $s_0 \in S$. It would then be important to compare the approach of generalized categories with feedback to the more standard approaches based on guarded recursion [20] and coalgebras [11, 35].

References

[1] Samson Abramsky. What are the fundamental structures of concurrency? We still don't know! *CoRR*, abs/1401.4973, 2014.

[2] Jirí Adámek, Stefan Milius, and Jiri Velebil. Elgot algebras. *Log. Methods Comput. Sci.*, 2(5), 2006.

[3] John C. Baez and Kenny Courser. Structured cospans. *CoRR*, abs/1911.04630, 2019.

[4] Jean Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, pages 1–77. Springer, 1967.

[5] Nick Benton and Martin Hyland. Traced premonoidal categories. *RAIRO Theor. Informatics Appl.*, 37(4):273–299, 2003.

[6] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories - The Equational Logic of Iterative Processes.* EATCS Monographs on Theoretical Computer Science. Springer, 1993.

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

[8] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. The Calculus of Signal Flow Diagrams I: Linear Relations on Streams. *Information and Computation*, 252:2–29, 2017.

[9] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. A connector algebra for P/T nets interactions. In *Concurrency Theory (CONCUR '11)*, volume 6901 of *LNCS*, pages 312–326. Springer, 2011.

[10] Aurelio Carboni and Robert F. C. Walters. Cartesian Bicategories I. *Journal of pure and applied algebra*, 49(1-2):11–32, 1987.

[11] Ranald Clouston, Aleš Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. Programming and reasoning with guarded recursion for coinductive types, 2015.

[12] William Henry Eccles and Frank Wilfred Jordan. Improvements in ionic relays. *British patent number: GB 148582*, 1918.

[13] Calvin C. Elgot. Monadic computation and iterative algebraic theories. In *Studies in Logic and the Foundations of Mathematics*, volume 80, pages 175–230. Elsevier, 1975.

[14] Brendan Fong. Decorated cospans. *Theory and Applications of Categories*, 30(33):1096–1120, 2015.

[15] Alessandro Gianola, Stefano Kasangian, Desiree Manicardi, Nicoletta Sabadini, Filippo Schiavio, and Simone Tini. CospanSpan(Graph): a compositional description of the heart system. *Fundam. Informaticae*, 171(1-4):221–237, 2020.

[16] Alessandro Gianola, Stefano Kasangian, Desiree Manicardi, Nicoletta Sabadini, and Simone Tini. Compositional modeling of biological systems in CospanSpan(Graph). In *Proc. of ICTCS 2020.* CEUR-WS, To appear.

[17] Alessandro Gianola, Stefano Kasangian, and Nicoletta Sabadini. Cospan/Span(Graph): an Algebra for Open, Reconfigurable Automata Networks. In Filippo Bonchi and Barbara König, editors, *7th Conference on Algebra and Coalgebra in Computer Science, CALCO 2017, June 12-16, 2017, Ljubljana, Slovenia*, volume 72 of *LIPIcs*, pages 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[18] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[19] Jean-Yves Girard. Towards a geometry of interaction. *Contemporary Mathematics*, 92(69-108):6, 1989.

[20] Sergey Goncharov and Lutz Schröder. Guarded traced categories. In *International Conference on Foundations of Software Science and Computation Structures*, pages 313–330. Springer, 2018.

[21] Masahito Hasegawa. Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi. pages 196–213. Springer Verlag, 1997.

[22] Masahito Hasegawa. The uniformity principle on traced monoidal categories. In Richard Blute and Peter Selinger, editors, *Category Theory and Computer Science, CTCS 2002, Ottawa, Canada, August 15-17, 2002*, volume 69 of *Electronic Notes in Theoretical Computer Science*, pages 137–155. Elsevier, 2002.

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

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

[25] Rudolf Emil Kalman, Peter L. Falb, and Michael A. Arbib. *Topics in mathematical system theory*, volume 1. McGraw-Hill New York, 1969.

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

[27] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. Span(Graph): A Categorial Algebra of Transition Systems. In Michael Johnson, editor, *Algebraic Methodology and Software Technology, 6th International Conference, AMAST '97, Sydney, Australia, December 13-17, 1997, Proceedings*, volume 1349 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1997.

[28] 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.

[29] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. A formalization of the IWIM model. In *International Conference on Coordination Languages and Models*, pages 267–283. Springer, 2000.

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

[31] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. A Process Algebra for the Span(Graph) Model of Concurrency. *arXiv preprint arXiv:0904.3964*, 2009.

[32] Stephen Lack. Composing PROPs. *Theory and Applications of Categories*, 13(9):147–163, 2004.

[33] S. J. Mason. Feedback Theory - Some properties of signal flow graphs. *Proceedings of the IRE*, 41(9):1144–1156, 1953.

[34] George H. Mealy. A method for synthesizing sequential circuits. *The Bell System Technical Journal*, 34(5):1045–1079, 1955.

[35] Stefan Milius and Tadeusz Litak. Guard your daggers and traces: Properties of guarded (co-) recursion. *Fundamenta Informaticae*, 150(3-4):407–449, 2017.

[36] Robert Rosebrugh, Nicoletta Sabadini, and Robert F. C. Walters. Generic commutative separable algebras and cospans of graphs. *Theory and applications of categories*, 15(6):164–177, 2005.

[37] Nicoletta Sabadini, Filippo Schiavio, and Robert F. C. Walters. On the geometry and algebra of networks with state. *Theor. Comput. Sci.*, 664:144–163, 2017.

[38] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

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

[40] Paweł Sobociński. A non-interleaving process calculus for multi-party synchronisation. In *2nd Interaction and Concurrency Experience: Structured Interactions, (ICE 2009)*, volume 12 of *EPTCS*, 2009.

[41] Paweł Sobociński. Representations of Petri net interactions. In *Concurrency Theory, 21th International Conference, (CONCUR 2010)*, number 6269 in LNCS, pages 554–568. Springer, 2010.