Full Abstraction for Signal Flow Graphs

Filippo Bonchi
ENS Lyon, U. Lyon, CNRS, INRIA
filippo.bonchi@ens-lyon.fr

Paweł Sobociński
U. Southampton, UK
ps@ecs.soton.ac.uk

Fabio Zanasi
ENS Lyon, U. Lyon, CNRS, INRIA
fabio.zanasi@ens-lyon.fr

Abstract

Network theory uses the string diagrammatic language of monoidal categories to study graphical structures formally, eschewing specialised translations into intermediate formalisms. Recently, there has been a concerted research focus on developing a network theoretic approach to signal flow graphs, which are classical structures in control theory, signal processing and a cornerstone in the study of feedback. In this approach, signal flow graphs are given a relational denotational semantics in terms of formal power series.

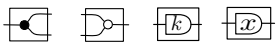
Thus far, the operational behaviour of such signal flow graphs has only been discussed at an intuitive level. In this paper we equip them with a structural operational semantics. As is typically the case, the purely operational picture is too concrete – two graphs that are denotationally equal may exhibit different operational behaviour. We classify the ways in which this can occur and show that any graph can be *realised* – rewritten, using the graphical theory, into an *executable* form where the operational behavior and the denotation coincides.

Categories and Subject Descriptors D.3.1 [*Formal Definitions and Theory*]: Semantics; F.3.2 [*Semantics of Programming Languages*]: Algebraic approaches to semantics

Keywords Signal Flow Graphs, String Diagrams, PROPs, Structural Operational Semantics, Full Abstraction

1. Introduction

Signal flow graphs (SFGs) are foundational structures in control theory and signal processing studied since at least the 1950s [23]. They can be constructed from small set of basic components (displayed below) and feedbacks.



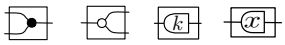
(1)

Signals, which take values over a field k, flow from left to right. The leftmost component *duplicates* the signal, the second *sums* the two signals arriving on the left and the third *multiplies* the signal by a scalar $k \in \mathsf{k}$. The rightmost one is a *delay*: when a sequence of signals $k_0, k_1, k_2, \ldots$ arrives on the left, it outputs the sequence $0, k_0, k_1 \ldots$ It can thus be thought as a synchronous one place buffer initialised with 0.

POPL '15, January 15–17, 2015, Mumbai, India.

http://dx.doi.org/10.1145/2676726.2676993

A simple mathematical meaning can be given to those SFGs where feedbacks pass through (at least) one delay component. It is well known (see e.g. [21]) that SFGs with this restriction, one input and one ouput port denote so-called rational linear functions. In traditional approaches, however, SFGs are not treated as interesting mathematical structures per se: formal analyses typically mean the introduction of latent variables and translations into systems of linear equations— although, more recently, they have also attracted the use of coalgebraic tools [4, 26]. This paper, instead, follows the series of recent works [3, 5, 7, 14, 31] where SFGs are understood as structures known as *string diagrams* and studied as mathematical objects of interest in their own right—this approach is known as *network theory* [2]. The majority of the attention so far has been focused on what we call the *denotational semantics*: differently from the classical approach, string diagrams, in general, give rise to *linear relations* rather than functions. The string diagrams that are considered are not restricted by any side conditions on feedbacks, being all those diagrams generated from the basic components (1), together with their duals:



(2)

Intuitively, in (2) the signal flows from right to left. This means that diagrams constructed using both components in (1) and (2) have no univocal flow direction and require a relational model.

Network theory brings fundamentally new ingredients to the field of signal flow graphs. First, the relational semantics is a *compositional* account of their behavior that enjoys a sound and complete axiomatisation independently discovered in [5, 7] and [3]. Second, the axiomatisation has uncovered a rich underlying mathematical playground – featuring two Hopf algebras and two Frobenius algebras – which also reveals connections with quantum phenomena [3, 6, 31]. Third, it has resulted in a subtle re-evaluation of *causality* as a central ingredient of SFGs. In 1953 Mason [23] wrote: "flow graphs differ from electrical network graphs in that their branches are directed. In accounting for branch directions it is necessary to take an entirely different line of approach from that adopted in electrical network topology." Instead, our results suggest that *direction of signal flow is not a primitive notion*: this argument has been made informally already in [3, 5] but is rigorously shown at the end of this paper (Section 6). Similar ideas are prominent in the *behavioural approach* in control theory [30].

In this paper, we introduce *operational semantics* to the network theoretic accounts of signal flow graphs: we show that string diagrams can be thought of as terms of a process calculus and executed as state machines. For this reason we shall call our string diagrams *circuit diagrams* or simply *circuits*. Reconciling the operational perspective with the established denotational model turns out to be quite subtle. Indeed, the denotational semantics is in a sense too abstract: finite computations that reach *deadlocks* are ignored. Such deadlocks can arise for instance when components of (1) are composed with the those of (2) and, intuitively, the signal

flows from the left and right toward the middle. For an example, consider the circuit below on the left.

(3)

In a first step, the signals arriving from left and right are stored in the two buffers. Then, the stored values are compared in the middle of the circuit: if they do not agree then the computation gets stuck. The circuit on the right features another problem, which we call *initialisation*. Intuitively, the flow goes from the middle toward left and right. All its computations are forced to start by emitting on the left and on the right the value 0 which is initially stored in the two buffers. The two circuits are denotationally equivalent, but their operational behaviour can be obviously distinguished: the leftmost does not have initialisation and the rightmost cannot deadlock.

Deadlock and initialisation are dual problems at the heart of the mismatch of operational and denotational semantics. We show that circuits in *cospan form*, namely circuits built from components in (1) followed by those in (2) (like the leftmost circuit in (3)), are free from initialisation. Instead, circuits in *span form*, i.e., those built from components in (2) followed by (1) (like the rightmost in (3)) are free from deadlock. This is interesting because the equational theory developed in [3, 5] asserts that any circuit is equivalent to both one in cospan and one in span form. This duality of deadlock and initialisation helps us in proving a *full abstraction* result: for those circuits that are free from both deadlock and initialisation, the operational and the denotational semantics agree.

Our second main theorem is a *realisability* result: for any denoted behaviour there exists some circuit that properly, without deadlocks or initialisation, *realises* it. The key for the proof is the fact that any circuit in [3, 5] is equivalent (according to the denotational semantics) to a signal flow graph up to some "rewiring". This result allow us to impose a syntactic restriction to our circuits to guarantee deadlock and initialisation freedom. By virtue of the full abstraction results we can thus safely use the axiomatization of [3, 5] to reason about the operational behaviour of these circuits.

Summarising, the main results of this paper are:

- a structural operational semantics for the network-theoretic account of SFGs;
- a full abstraction theorem relating the operational and the denotational semantics previously introduced in [3, 5];
- a realisability theorem: every behaviour can be implemented by a circuit without deadlock and initialisation;
- a formal explanation of the fact that direction of flow is a derivative notion.

Related work. String diagrams originally came to the fore in the study of monoidal categories because they clear away swathes of cumbersome coherence bureaucracy, thereby dramatically simplifying algebraic arguments: in particular, they are useful for characterising *free* monoidal categories [16, 18, 27].

In this paper we work with particular symmetric monoidal categories, called PROPs [19, 22] (PROduct-and-Permutation categories). PROPs are a useful setting for the study of string diagrams and especially monoidal theories—Lack's theory of composing PROPs [19] was used to derive the axiomatisation in [5, 7]. PROPs have also recently been used by computer scientists: Lafont's study of boolean circuits [20], Bruni, Montanari, Plotkin, and Terreni [8] have used them to give an alternative presentation of Milner's bigraphs while Fiore and Campos [13] presented a theory of directed acyclic graphs. Our operational semantics is related to Katis, Sabadini and Walters' [17] Span(Graph) algebra of transition systems and the algebra of connectors of Bruni, Lanese and Montanari [9]. String diagrams are increasingly used by computer scientists: for instance we mention Pavlovic's monoidal computer [24, 25] where they are employed to study classical notions of computability and computational complexity.

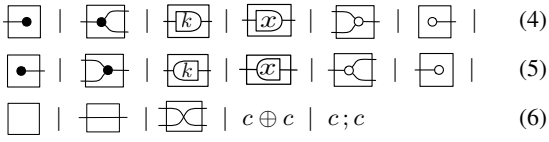
The interplay of Hopf algebras and Frobenius algebras, at the core of the axiomatisation of the denotational semantics, appeared first in the work of Coecke, Duncan and Kissinger [11, 12] on the ZX-calculus, used in the study of quantum circuits. Similar algebraic interactions emerged in the study of Petri nets [28] and in string-diagrammatic theories of asynchronous circuits [15].

Structure. In §2 we introduce the operational semantics. In §3 we recall the denotational semantics from [3, 5] and we prove full abstraction in §4. In §5 we prove the realisability theorem and in §6 we consider a directed syntax in order to capture classical SFGs.

Notational conventions. $\mathbb{C}[a, b]$ is the set of arrows from a to b in a small category $\mathbb{C}$. Composition of $f\colon a \to b$, $g\colon b \to c$ is written $f\,;g\colon a \to c$. When $\mathbb{C}$ is monoidal, $\oplus$ is the monoidal product.

2. The Signal Flow Calculus: Syntax and Operational Semantics

Here we give the syntax and the structural operational semantics of a simple process calculus, to which we shall refer to as the *Signal Flow Calculus*. Fix an arbitrary field k. The syntax, given below, does not feature binding nor primitives for recursion, while k ranges over k.

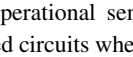
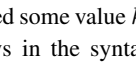
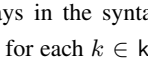
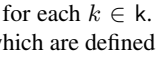
$c ::=$ 

(4)

(5)

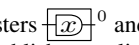
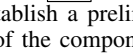
(6)

A *sort* is a pair (n, m), with $n, m \in \mathbb{N}$. We shall consider only terms that are sortable, according to the rules of Fig. 1. A simple inductive argument confirms uniqueness of sorting: if $c : (n, m)$ and $c : (n', m')$ then $n = n'$ and $m = m'$. We will refer to sortable terms as *circuits* since, intuitively, a term $c : (n, m)$ represents a circuit with n ports on the left and m ports on the right. The wires carry elements of a field k.

The operational semantics is a transition system with states augmented circuits where each *delay* component ( and ) is assigned some value $k \in \mathsf{k}$. Thus *states* are obtained by replacing the delays in the syntax specification with *registers*  and  for each $k \in \mathsf{k}$. As for circuits, we only consider sortable states, which are defined by adding

to the rules in Fig. 1.

Structural inference rules for operational semantics are given in Fig. 2 where we use strings of length n to represents vectors in k^n. So, the empty string stands for $(\,)$, the only vector of k^0, and $v = k_1 \dots k_n$ for the column vector $\begin{pmatrix} k_1 \\ \vdots \\ k_n \end{pmatrix}$ in k^n.

If state $s : (n, m)$ is the source of a transition $\xrightarrow[w]{v} t$ then t is also a state with sort (n, m) and v and w are strings representing vectors of k^n and k^m, respectively. Intuitively, $s \xrightarrow[w]{v} t$ means that s can become t whenever the signals on the n ports on the left agree with v and the signals on the m ports on the right agree with w. Each circuit c then yields a transition system with a chosen *initial state* s_0 of c, obtained by replacing the delays  and  in c with registers and containing 0.

To establish a preliminary intuition, we can consider circuits built up of the components in (4) as taking signals from the left

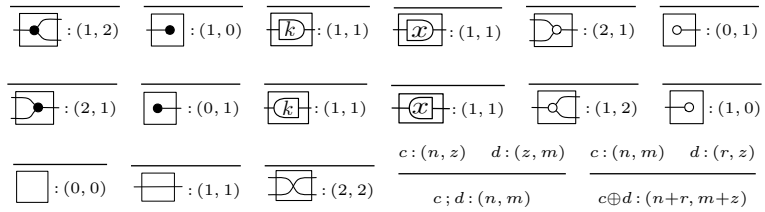


Figure 1. Sort inference rules.

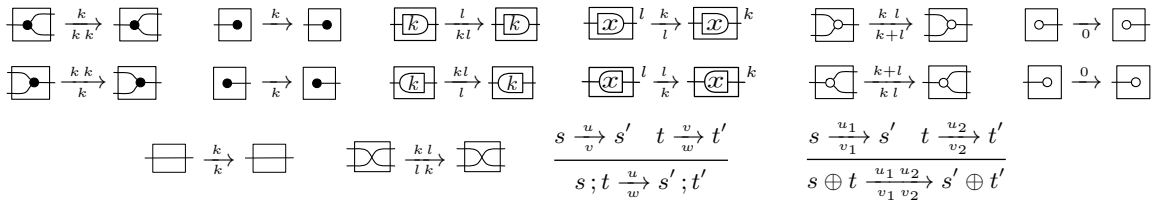


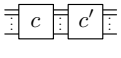
Figure 2. Structural rules for operational semantics, with k, l ranging over k and u, v, w vectors of elements of k of the appropriate size.

boundary to the right: thus is a *copier*, duplicating the signal arriving on the left; accepts any signal on the left and discards it, producing nothing on the right; is an *adder* that takes two signals on the left and emits their sum on the right, and constantly emits the signal 0 on the right; is an *amplifier*, multiplying the signal on the left by the scalar $k \in \mathsf{k}$. Finally, is a *delay*, a synchronous one place buffer initialised with 0.

The terms of row (5) are those of row (4) "reflected about the y-axis". Their behaviour is symmetric—indeed, here it is helpful to think of signals flowing from right to left.

In row (6), is a *twist*, swapping two signals, is the empty circuit and is the *identity* wire: the signals on the left and on the right ports are equal. Terms can be combined by two binary operators: sequential ; and parallel $\oplus$ composition.

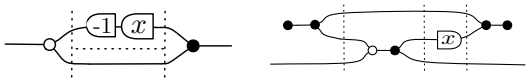
In the syntax specification we purposefully used a graphical rendering of the components. Indeed, we will seldom write terms in the traditional way and instead represent them as 2-dimensional diagrams. We adopt the following common convention:

$c\,;c'$ is drawn  $c \oplus c'$ is drawn  .

A *computation* of a circuit c, is a (possibly infinite) path $s_0 \xrightarrow[w_0]{v_0} s_1 \xrightarrow[w_1]{v_1} \dots$ in the transition system of c, starting from its initial state s_0. When c has sort $(n,\, m)$, each v_i and w_i consist of strings over k, say $k_{i1} \dots k_{in}$ and $l_{i1} \dots l_{im}$, respectively. The *trace* of a computation $s_0 \xrightarrow[w_0]{v_0} s_1 \xrightarrow[w_1]{v_1} \dots$ is then a pair of vectors $\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}, \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}$ where $\alpha_j = k_{0j} k_{1j} \dots$ and $\beta_j = l_{0j} l_{1j} \dots$. Occasionally we will use the notation $(\overrightarrow{\alpha}, \overrightarrow{\beta})$ for such a pair and, to make the notation lighter, we will write $\alpha_j = k_0 k_1 \dots$ and $\beta_j = l_0 l_1 \dots$. Moreover, with $\alpha_j(i)$ and $\beta_j(i)$ we will denote the i-th elements of α_j and β_j.

Note that in a computation of length z, all α_j, β_j have length z, while for an infinite computation all α_j, β_j are infinite. In the former case, we say that a trace is finite, in the latter that it is infinite. We use $ft(c)$ to denote the set of all finite traces of c and $it(c)$ for the set of all infinite ones.

Example 1. Consider the two circuits below.



The first is a graphical representation of the term

$$c_1 = (\,; ((\,; \,) \oplus)) \,;$$

the second of the term

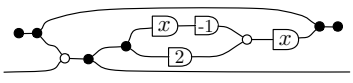
$$c_2 = ((\,; \,) \oplus)\,; (\oplus (\,; \,))$$
$$\,; (((\oplus) \oplus)\,; ((\,; \,) \oplus))$$

Note that, according to our intuition, in the leftmost circuit the signal flows from right to left, while the rightmost, the signal flows from left to right – indeed, the terms ; and ; serve as "bent identity wires" which allow us to form a feedback loop. Let $c_1[k]$ and $c_2[k]$ represent the states of c_1 and c_2, with k denoting the value at the register. The rules of Fig. 2 yield the computation

$$c_i[0] \xrightarrow[1]{1} c_i[1] \xrightarrow[1]{0} c_i[1] \xrightarrow[1]{0} c_i[1] \cdots$$

for $i \in 0, 1$, which yields the trace $(1000\dots), (1111\dots)$. In fact, as we shall show via a sound and complete axiomatisation, despite of the signal intuitively flowing in different directions, the two circuits have the same observable behaviour.

A slightly more involved example is given below.



We leave the reader to write down a term that is represented by the diagram above: call it c_3 and let $c_3[k_1, k_2]$ represent the state where the two registers, reading from from left to right, have values k_1 and k_2. Then, the operational semantics allows us to derive the

following computation

$$c_3[0,0] \xrightarrow{\frac{1}{1}} c_3[1,2] \xrightarrow{\frac{0}{2}} c_3[2,3] \xrightarrow{\frac{0}{3}} c_3[3,4] \xrightarrow{\frac{0}{4}} \cdots$$

that yields the trace $(1000\ldots, 1234\ldots)$.

Circuits Diagrams. In the first two diagrams of Example 1 we used dotted lines to ease the passage from each diagram to the corresponding syntactic term. Indeed, it is clear that syntax carries more information than the diagrammatic notation (e.g. associativity). From the point of view of operational behaviour, however, this extra information is irrelevant and is conveniently discarded by the graphical notation: we will never again blemish our diagrams with dotted lines.

Remark 1. Checking that this "forgetting" is sound amounts to verifying that for any circuits c_1, c_2, c_3, c_4, the following circuits (when sortable) yield isomorphic transition systems:

- $(c_1 \,;\, c_2) \,;\, c_3$ and $c_1 \,;\, (c_2 \,;\, c_3)$,
-  $;\, c_1$, c_1 and $c_1 \,;$,
- $(c_1 \oplus c_2) \oplus c_3$ and $c_1 \oplus (c_2 \oplus c_3)$,
-  $\oplus\, c_1$, c_1 and $c_1 \oplus$,
- $(c_1 \,;\, c_3) \oplus (c_2 \,;\, c_4)$ and $(c_1 \oplus c_2) \,;\, (c_3 \oplus c_4)$.

Of course, the reader will notice a close connection with the axioms of monoidal categories — we make explicit use of this fact below. In fact we will use *symmetric* monoidal categories (SMCs) of a specific kind, namely PROPs [19, 22]: a PROP (product and permutation category) is a strict SMC with objects the natural numbers, where $\oplus$ on objects is by addition. Morphisms between PROPs are strict symmetric monoidal functors that act as identity on objects: PROPs and their morphisms form the category $\mathbf{PROP}$.

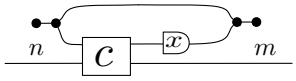
Definition 1. The PROP Circ of *circuit diagrams* is defined as:

- arrows $n \to m$ are circuit terms of sort (n, m) quotiented by the axioms of symmetric monoidal categories (see *e.g.* [27]). Composition ; and monoidal product $\oplus$ of circuits are given by the syntax operations in (6).
- The identities are $id_0 :=$  and $id_{n+1} := id_n \oplus$  . The symmetries $\sigma_{n,m} \colon n+m \to m+n$ are defined in the obvious way starting from $\sigma_{1,1} :=$  . For instance, $\sigma_{2,3}$ is (up-to the axioms of SMCs) the circuit below.

Observe that all the axioms of SMCs are sound (à la Remark 1) with respect to the operational semantics given in Figure 2: for two state terms c and d representing the same state diagram, the corresponding transition systems are isomorphic. This means that there is not any problem in reasoning up to the axioms of SMCs. For this reason, in the rest of the paper we shall refer to circuits diagrams and state diagrams just as *circuits* and *states*, purposefully blurring the line between diagrams and traditional syntax.

We identify two sub-PROPs of Circ: $\mathsf{C}\overrightarrow{\mathsf{irc}}$ has as arrows only those circuits in Circ that are built from the components of (4) and (6) and $\mathsf{C}\overleftarrow{\mathsf{irc}}$ only those circuits built from the components of (5) and (6). The notation emphasises that for circuits in $\mathsf{C}\overrightarrow{\mathsf{irc}}$, signal flow is from left to right, and in $\mathsf{C}\overleftarrow{\mathsf{irc}}$ from right to left. Formally, observe that $\mathsf{C}\overleftarrow{\mathsf{irc}}$ is the opposite category of $\mathsf{C}\overrightarrow{\mathsf{irc}}$: any circuit of $\mathsf{C}\overleftarrow{\mathsf{irc}}$ can be seen as one of $\mathsf{C}\overrightarrow{\mathsf{irc}}$ reflected about the y-axis. We also remark that Circ is the coproduct in $\mathbf{PROP}$ of $\mathsf{C}\overrightarrow{\mathsf{irc}}$ and $\mathsf{C}\overleftarrow{\mathsf{irc}}$.

Beyond $\mathsf{C}\overrightarrow{\mathsf{irc}}$ and $\mathsf{C}\overleftarrow{\mathsf{irc}}$, we can identify another class of circuits of Circ which adhere to the classical notion of *signal flow graph* (see *e.g.* [23]). In these circuits, the signal flows from left to right, like in $\mathsf{C}\overrightarrow{\mathsf{irc}}$, but with the possibility of having *feedback loops*, provided that these pass through at least one delay. Formally, this amounts to defining, for each n and m, an assignment $\mathsf{Tr}(\cdot) \colon \mathsf{Circ}[n+1, m+1] \to \mathsf{Circ}[n, m]$ mapping a circuit $c \colon n+1 \to m+1$ into the n-to-m circuit below:



In the picture above and in the sequel, we use the shorthand notation  for a circuit of the form id_z. The intuition is that $\mathsf{Tr}(\cdot)$ equips the circuit c with a feedback loop carrying the signal from its topmost right to its topmost left port.

Signal flow graphs form a PROP SF, which is the sub-PROP of Circ inductively defined as follows:

- if $c \in \mathsf{C}\overrightarrow{\mathsf{irc}}[n, m]$, then $c \in \mathsf{SF}[n, m]$
- if $c \in \mathsf{SF}[n+1, m+1]$, then $\mathsf{Tr}(c) \in \mathsf{SF}[n, m]$
- if $c_1 \in \mathsf{SF}[n, z]$ and $c_2 \in \mathsf{SF}[z, m]$, then $c_1 ; c_2 \in \mathsf{SF}[n, m]$
- if $c_1 \in \mathsf{SF}[n, m]$ and $c_2 \in \mathsf{SF}[r, z]$, then $c_1 \oplus c_2 \in \mathsf{SF}[n+r, m+z]$.

For instance, the second and third circuit of Example 1 are in SF, whereas the first one is in $\mathsf{C}\overleftarrow{\mathsf{irc}}$.

Remark 2. The rules of Figure 2 describe the step-by-step evolution of state machines without relying on a fixed flow orientation. This operational semantics is not meant to be executable for all circuits: the rule for sequential composition implicitly quantifies existentially on the middle value v, resulting in potentially unbounded non-determinism. However, for circuits where flow directionality can be assigned, like the class SF above, existential quantification becomes deterministic subject to a choice of inputs to the circuit at each step of evaluation. We will see in Section 6 that any circuit can be transformed into this form, where the valid transformations are those allowed by the equational theory, presented below.

3. Denotational Semantics

Here we define the denotational domain of interpretation for circuits; all the results in this section are proven in or immediately follow from [5, 7]. We begin by recalling some background.

A formal Laurent series (fls) is a function $\sigma \colon \mathbb{Z} \to \mathsf{k}$ for which there exists $i \in \mathbb{Z}$ such that $\sigma(j) = 0$ for all $j < i$. The *degree* of σ is the smallest $d \in \mathbb{Z}$ such that $\sigma(d) \neq 0$. We write σ as $\ldots, \sigma(-1), \underline{\sigma(0)}, \sigma(1), \ldots$ with position 0 underlined, or as formal sum $\sum_{i=d}^{\infty} \sigma(i) x^i$. With the latter notation, we define the sum and product of $\sigma = \sum_{i=d}^{\infty} \sigma(i) x^i$ and $\tau = \sum_{i=e}^{\infty} \tau(i) x^i$ as

$$\sigma + \tau \;=\; \sum_{i=min(d,e)}^{\infty} \big(\sigma(i) + \tau(i)\big) x^i \tag{7}$$

$$\sigma \cdot \tau \;=\; \sum_{i=d+e}^{\infty} \Big(\sum_{k+j=i} \sigma(j) \cdot \tau(k) \Big) x^i \tag{8}$$

The units for $+$ and $\cdot$ are $\ldots 0, \underline{0}, 0 \ldots$ and $\ldots 0, \underline{1}, 0 \ldots$. Fls form a field $\mathsf{k}((x))$, where the inverse σ^{-1} of fls σ with degree d is:

$$\sigma^{-1}(i) = \begin{cases} 0 & \text{if } i < -d \\ \sigma(d)^{-1} & \text{if } i = -d \\ \frac{\sum_{i=1}^{n} \left(\sigma(d+i) \cdot \sigma^{-1}(-d+n-i)\right)}{-\sigma(d)} & \text{if } i = -d+n \text{ for } n > 0 \end{cases} \tag{9}$$

A *formal power series* (fps) is a fls with degree $d \geq 0$. By (7) and 8 fps are closed under $+$ and $\cdot$, but not under inverse: it is immediate

from (9) that σ^{-1} is a fps iff σ has degree $d = 0$. Therefore fps form a ring which we denote by $\mathsf{k}[[x]]$.

Other algebras of interest are the ring $\mathsf{k}[x]$ of *polynomials* and its field of *fractions* $\mathsf{k}(x)$. A polynomial $k_0 + k_1 x + \cdots + k_n x^n$ can also be regarded as the fps $\sum_{i=0}^{\infty} k_i x^i$ with $k_i = 0$ for all $i > n$. Instead, in order to express fractions we need the full generality of fls: there is a unique field morphism mapping $k \in \mathsf{k}(x)$ into the fls $\ldots 0, \underline{k}, 0 \ldots$ and the indeterminate x into $\ldots 0, \underline{0}, 1, 0 \ldots$. This morphism will map, in particular, the fraction $\frac{1}{x}$ into the fls $\ldots 0, 1, \underline{0}, 0, 0 \ldots$.

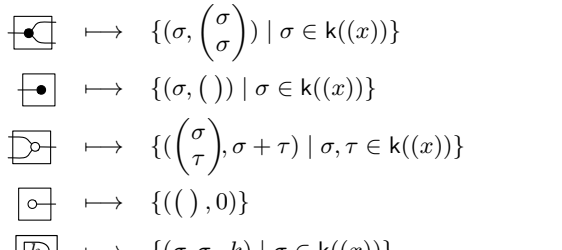
In Section 4.1, we will use polynomials to encode finite sequences, and fps for streams. In this perspective, fls can be thought of as sequences with an infinite future, but a "finite past". Note that, differently from polynomials, fractions can express infinite sequences. For instance, $\frac{1}{(1-x)^2}$ denotes (along the field morphism introduced above) the fls $\ldots, 0, 0, \underline{1}, 2, 3, \ldots$

Definition 2. Let $\mathsf{Rel}_{\mathsf{k}((x))}$ be the following PROP:

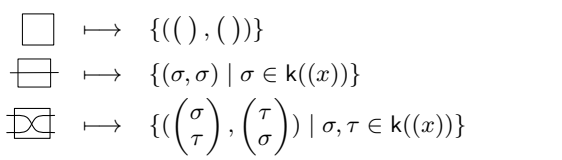
- arrows $n \to m$ are subsets of $\mathsf{k}((x))^n \times \mathsf{k}((x))^m$.
- composition is relational: given $G = \{(u, v) \mid u \in \mathsf{k}((x))^n, v \in \mathsf{k}((x))^z\}$ and $H = \{(v, w) \mid v \in \mathsf{k}((x))^z, w \in \mathsf{k}((x))^m\}$, their composition is $\{(u, w) \mid \exists v.(u, v) \in G \wedge (v, w) \in H\}$.
- $G \oplus H = \{\left(\begin{pmatrix} u \\ u' \end{pmatrix}, \begin{pmatrix} v \\ v' \end{pmatrix}\right) \mid (u, v) \in G, (u', v') \in H\}$.
- The symmetries $n \to n$ are induced by bijections of finite sets: $\rho : n \to n$ is associated with the subset $\{(v, w) \mid v, w \in \mathsf{k}((x))^n,\ v_i = w_{\rho i}\}$.

$\mathsf{Rel}_{\mathsf{k}((x))}$ serves as the domain for the denotational semantics.

Definition 3. The PROP morphism $[\![\cdot]\!] : \mathsf{Circ} \to \mathsf{Rel}_{\mathsf{k}((x))}$ is inductively defined on circuits as follows. For the components in (4)

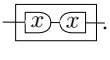


where 0, k and x denote fls. The semantics of components in (5) is symmetric, e.g.  is mapped to $\{((\,), \sigma) \mid \sigma \in \mathsf{k}((x))\}$. For (6)



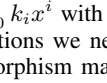
$$c_1 \oplus c_2 \longmapsto [\![c_1]\!] \oplus [\![c_2]\!] \qquad c_1 \,;\, c_2 \longmapsto [\![c_1]\!] \,;\, [\![c_2]\!]$$

Remark 3. It is easy to verify that for any circuit $c \in \mathsf{Circ}[n, m]$, $[\![c]\!]$ forms a subspace of $\mathsf{k}((x))^n \times \mathsf{k}((x))^m$, considered as a vector space over $\mathsf{k}((x))$. See [5] for more details.

Example 2. Consider the circuit  . We have that

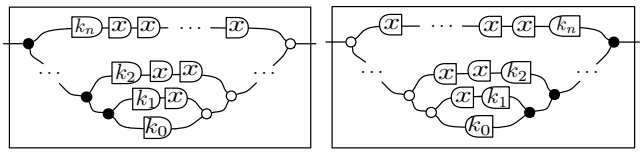
$$\begin{aligned} [\![\text{}]\!] &= [\![\text{  }]\!] \,;\, [\![\text{}]\!] \\ &= \{(\sigma, \sigma \cdot x) \mid \sigma \in \mathsf{k}((x))\} \,;\, \{(\sigma \cdot x, \sigma) \mid \sigma \in \mathsf{k}((x))\} \\ &= \{(\sigma, \sigma) \mid \sigma \in \mathsf{k}((x))\} \end{aligned}$$

which is equal to $[\![$  $]\!]$. Note that any fls $\ldots \sigma(-1), \underline{\sigma(0)}, \sigma(1) \ldots$ on the left of  is related to $\ldots \underline{\sigma(-1)}, \sigma(0), \sigma(1) \ldots$ on its right and this is in turn related to $\ldots \sigma(-1), \underline{\sigma(0)}, \sigma(1) \ldots$ on the right of  . The circuit is thus the inverse of : while delays σ, accelerates it.

Similarly, consider a circuit  . Its semantics is the composite of $[\![$ $]\!]$ (pairing σ with $\sigma \cdot k$) and $[\![$ $]\!]$ (pairing $k \cdot \sigma$ with σ): if $k \neq 0$, we can see it as first multiplying and then dividing σ by k. Thus for $k \neq 0$ and have the same denotation.

3.1 Equational Theory

The equivalence induced by the denotational semantics is axiomatized in Figures 3, 4 and 5. There, p, p_1, p_2 range over $\mathsf{k}[x]$ and q over $\mathsf{k}[x] \setminus \{0\}$. Given a polynomial $p = k_0 + k_1 x + k_2 x^2 + \cdots + k_n x^n$,  and are notation for the circuit on the left and on the right respectively:



Observe that components , and , become specific cases of , respectively. Instead the notation indicates both the circuits and : indeed, they are provably equal from the other axioms (for instance, using (I5) and (A5)).

Let $\overset{\mathbb{IH}}{=}$ be the smallest congruence on circuits generated by these axioms and $\mathbb{IH}$ the PROP obtained by quotienting Circ by $\overset{\mathbb{IH}}{=}$.

Theorem 1. Let c, d be circuits in Circ. Then $[\![c]\!] = [\![d]\!]$ iff $c \overset{\mathbb{IH}}{=} d$.

The equational theory of $\mathbb{IH}$ equips circuits with a very rich algebraic structure, that can be presented in a modular way:

- Figure 3 describe the interaction of components in (4). (A1)-(A3) and (A6)-(A8) impose a commutative monoid and a cocommutative comonoid structure on the components  , and , respectively. Together they form a bialgebra, by laws (A9)-(A12). Instead, (A4), (A5), (A17), (A18) describe the behavior of derived components of shape , saying in particular that they are compatible with the bialgebra structure. Finally, note that the bialgebra is in fact an Hopf algebra, with antipode the circuit .

  It is convenient to fix this "module" of $\mathbb{IH}$ as a sub-PROP, which we call $\mathbb{HA}$ because of the Hopf algebra structure. It can be equivalently defined as the PROP having as arrows the circuits of $\mathsf{Circ}^{\rightarrow}$ quotiented by the equations in Figure 3.
- The algebraic structure described above is dualised for components in (5), in the sense that the axioms (An)* in Figure 4 are exactly those (An) of Figure 3 reflected about the y-axis. This means that , satisfy the equations of commutative comonoids, , the ones of commutative monoids, and together they form an Hopf Algebra with antipode . The corresponding sub-PROP is $\mathbb{HA}^{op}$, that is, the opposite category of $\mathbb{HA}$: for a circuit $c \in \mathbb{HA}[n, m]$, we draw the corresponding one in $\mathbb{HA}^{op}[m, n]$ as c reflected about the y-axis. Equivalently, $\mathbb{HA}^{op}$ can be defined as the PROP having as arrows the circuits of $\mathsf{Circ}^{\leftarrow}$ quotiented by the equations in Figure 4.

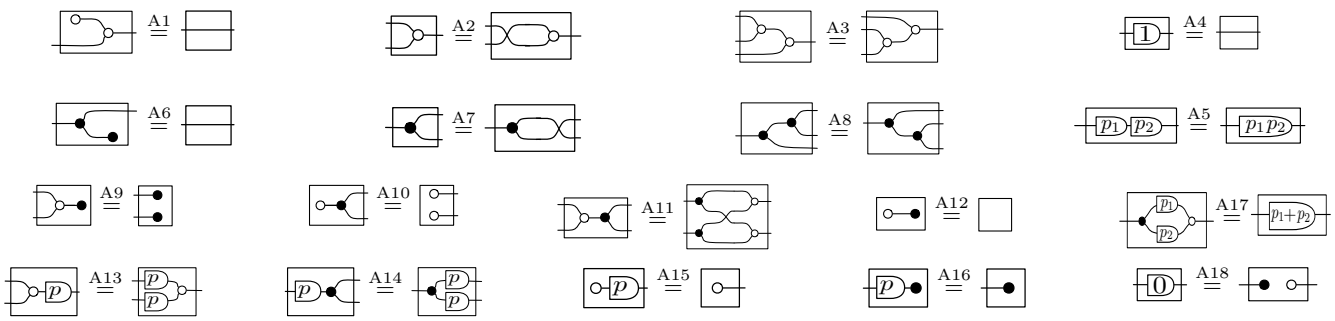


Figure 3. Interaction of components in (4)

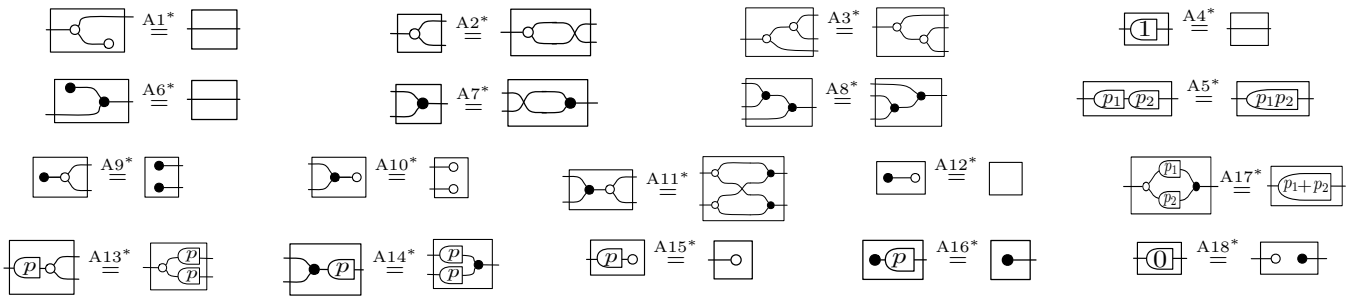


Figure 4. Interaction of components in (6)

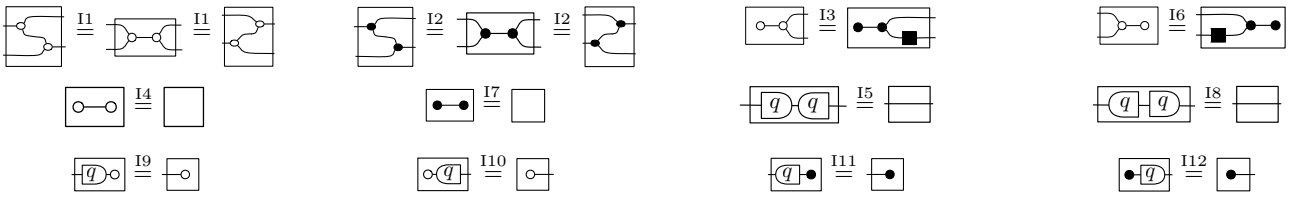


Figure 5. Interaction of components in (4) with the ones in (6)

- Finally, equations in Figure 5 describe the interaction of components in (4) with components in (5). This motivates the name $\mathbb{IH}$ for the theory of $\mathbb{I}$nteracting $\mathbb{H}$opf algebras. The axioms of interaction describe a separable Frobenius algebra [10] for both the white and the black family of components.

The subtheories $\mathbb{HA}$ and $\mathbb{HA}^{op}$ actually have an interesting life of their own: they characterise circuits with simple functional behaviors, in the following sense. Let $\mathsf{Mat}\,\mathsf{k}[x]$ be the PROP where arrows n to m are $m \times n$-matrices over $\mathsf{k}[x]$, composition $;$ is matrix multiplication, $A \oplus B$ is the matrix $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$ and the symmetries are the rearrangements of the rows of the identity matrix. Then $\mathbb{HA}$ is a presentation of $\mathsf{Mat}\,\mathsf{k}[x]$, that is, they are isomorphic as PROPs [5, 7]. Dually, $\mathbb{HA}^{op}$ is isomorphic to $\mathsf{Mat}\,\mathsf{k}[x]^{op}$.

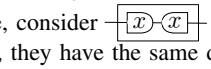
The circuits of $\mathbb{HA}$ and $\mathbb{HA}^{op}$ are not the only ones exhibiting a functional behavior. As shown in [5], by quotienting the PROP of signal flow graphs SF by the axioms of $\mathbb{IH}$, one obtains a PROP, hereafter referred to as $\mathbb{SF}$, that presents the PROP $\mathsf{Mat}\,\mathsf{k}\langle x\rangle$ of matrices over the *rationals*, i.e, fractions of polynomials $\frac{k_0+k_1x+k_2x^2\cdots+k_nx^n}{l_0+l_1x+l_2x^2\cdots+l_nx^n}$ where $l_0 \neq 0$. The definition of $\mathsf{Mat}\,\mathsf{k}\langle x\rangle$ is formally the same as the one of $\mathsf{Mat}\,\mathsf{k}[x]$, with the ring $\mathsf{k}\langle x\rangle$ of rationals replacing $\mathsf{k}[x]$.

The equational theory of $\mathbb{IH}$ allows us to factorise circuits of Circ in terms of those of $\mathsf{C}\overrightarrow{\mathsf{irc}}$ and $\mathsf{C}\overleftarrow{\mathsf{irc}}$. We say that $c \in \mathsf{Circ}[n,m]$ is in *cospan form* if it is of shape $c_1 ; c_2$, with $c_1 \in \mathsf{C}\overrightarrow{\mathsf{irc}}[n,z]$ and $c_2 \in \mathsf{C}\overleftarrow{\mathsf{irc}}[z,m]$ for some z. Dually, $d \in \mathsf{Circ}[n,m]$ is in *span form* if it is of shape $d_1 ; d_2$, with $d_1 \in \mathsf{C}\overleftarrow{\mathsf{irc}}[n,r]$ and $d_2 \in \mathsf{C}\overrightarrow{\mathsf{irc}}[r,m]$ for some r.

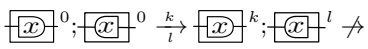
Proposition 1. For all circuits c of Circ, there exist circuits c' in span form and c'' in cospan form such that $c \overset{\mathbb{IH}}{=} c' \overset{\mathbb{IH}}{=} c''$.

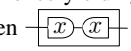
4. Full abstraction

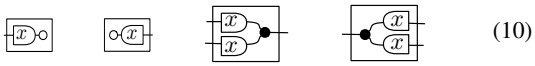
In this section we tackle the question of relating the operational and the denotational semantics of the Signal Flow Calculus, introduced in Section 2 and 3 respectively.

To this aim, an elementary observation is that the denotational semantics seems to be too coarse, since it abstracts away from the finite behaviours that might arise during the executions of the circuits. For example, consider  and  : as we have shown in Example 2, they have the same denotational semantics,

namely the set of all pairs (σ, σ) of fls. However, some computations of the former circuit can reach a *deadlock*. For instance, we can make a transition from the initial state with labels $k \neq l$, but there are no further possible transitions from the resulting state:



Such failures are not taken into account by the denotational semantics: intuitively, this only considers the successful computations, which are the ones yielding infinite traces. If we restrict to these situations, then  behaves exactly as the identity circuit  . Here are more examples of circuits that may reach a deadlocked state.

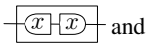
 (10)

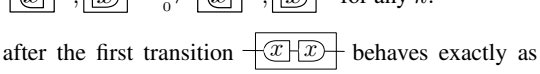
Our diagnosis is that problematic circuits are those in which internal components (in particular, the delays) have a conflicting design. Note that all the above examples are in *cospan form*, that is, they are of shape $c = c_1 \,;\, c_2$ with c_1 a circuit of $\mathsf{C}\overrightarrow{\mathsf{irc}}$ and c_2 one of $\mathsf{C}\overleftarrow{\mathsf{irc}}$. Intuitively, the signal in c is flowing from the left/right boundaries towards the middle, that is, the boundary shared by circuits c_1 and c_2.

According to our analysis, we can avoid deadlocks by considering instead circuits d in *span form*, i.e. $d = d_1 \,;\, d_2$ for d_1 in $\mathsf{C}\overleftarrow{\mathsf{irc}}$ and d_2 in $\mathsf{C}\overrightarrow{\mathsf{irc}}$. Circuits of this shape cannot deadlock since, intuitively, the signal is flowing from the middle boundary towards the left (transmitted by d_1) and the right (transmitted by d_2).

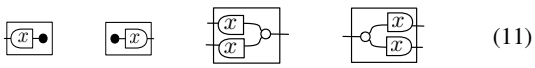
In order to formalize our observations, first we say that a circuit is *deadlock free* when none of its computations can reach a deadlock - namely, a state from which no transition is derivable. Then we have the following result.

Theorem 2. Circuits of Circ in span form are deadlock free.

Together with Proposition 1, this theorem asserts that, for each circuit of Circ, there exists an equivalent one in $\mathbb{IH}$ that is deadlock free. This could give us some hope of reconciling the operational and the denotational semantics but, unfortunately, also for some circuits in span form they do not agree: for instance,  and  have the same denotational semantics, but all the computation of the former are forced to start with $\xrightarrow[0]{0}$. Indeed

 for any k.

Note that after the first transition behaves exactly as  : in some sense, the former circuit exhibits a proper behaviour only after an initialisation step. To make this formal, we say that a circuit c is *initialisation free* if, whenever $s_0 \xrightarrow[0\ldots0]{0\ldots0} s_1$, then $s_1 = s_0$, where s_0 is the initial state of c. Other basic circuits that suffer from initialisation are displayed below.

 (11)

All problematic circuits above are in span form, meaning that they can be decomposed into $c_1 \,;\, c_2$, with c_1 in $\mathsf{C}\overleftarrow{\mathsf{irc}}$ and c_2 in $\mathsf{C}\overrightarrow{\mathsf{irc}}$. The intuition is that any delay in c_1 and c_2 sends the signal from the common middle boundary towards the outer boundaries, thus requiring a step in which the default value 0 of each delay is emitted before behaving properly. According to this analysis, such a situation is avoided when we can see all the delays as pointing towards the middle of the circuit. This leads to the following statement.

Theorem 3. Circuits of Circ in cospan form are initialisation free.

Theorems 2 and 3 suggest a duality between deadlock and initialisation, expressible in terms of span and cospan decompositions of circuits of Circ. In fact, this is reflected also by the displayed problematic circuits: the ones in (10) are dual to the ones in (11) in a precise sense, namely by changing the black/white colouring and the direction of delays[1]. Moreover, according to our analysis circuits with deadlocks have *more* behaviours (traces) than those prescribed by the denotational semantics, while circuits with initialisation have *fewer* behaviours. We would then expect circuits which are both deadlock and initialisation free to yield exactly the right amount of behavior: this will be the content of the next section, leading to the full abstraction result (Corollary 2).

4.1 Reconciling Observation and Denotation

Given a circuit c of Circ, we define its *observable behavior* $\langle c \rangle$ as the pair $(ft(c), it(c))$ of its finite and infinite traces. Like the denotational semantics $[\![\cdot]\!]$, also the observable behaviour $\langle \cdot \rangle$ can be expressed in a compositional way, as a PROP morphism from Circ to a certain target PROP that we are going to define below. In order to do that, we first observe that finite and infinite traces can be equivalently described in terms of polynomials and of fps respectively. Indeed, in a trace $(\overrightarrow{\alpha}, \overrightarrow{\beta})$ of length z, each sequence $\alpha_j = k_0 k_1 \ldots k_z$ and $\beta_j = l_0 l_1 \ldots l_z$ can be encoded as polynomials $k_0 x + k_1 x + \cdots + k_z x^z$ and $l_0 x + l_1 x + \cdots + l_z x^z$ respectively. Similarly, in an infinite trace $(\overrightarrow{\alpha}, \overrightarrow{\beta})$, each stream $\alpha_j = k_0 k_1 \ldots$ and $\beta_j = l_0 l_1 \ldots$ defines fps $\Sigma_{i=0}^{\infty} k_i x^i$ and $\Sigma_{i=0}^{\infty} j_i x^i$ respectively. We can then see $ft(c)$ as a relation between vectors of polynomials and $it(c)$ as a relation between vectors of fps.

On the base of this observation, we take $\mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$ as target of $\langle \cdot \rangle$. Here $\mathsf{Rel}_{\mathsf{k}[x]}$ and $\mathsf{Rel}_{\mathsf{k}[[x]]}$ are PROPs defined as $\mathsf{Rel}_{\mathsf{k}((x))}$ (Definition 2), but with $\mathsf{k}[x]$ and $\mathsf{k}[[x]]$ respectively in place of $\mathsf{k}((x))$. Arrows of $\mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$ from n to m are pairs (f, g) with $f \in \mathsf{Rel}_{\mathsf{k}[x]}[n, m]$ and $g \in \mathsf{Rel}_{\mathsf{k}[[x]]}[n, m]$. The following statement guarantees that $\langle \cdot \rangle$ is compositional:

Proposition 2. $\langle \cdot \rangle \colon \mathsf{Circ} \to \mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$ is a morphism of PROPs.

We have now all the ingredients to build a bridge between the domain of observations $\mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$ and the denotational domain $\mathsf{Rel}_{\mathsf{k}((x))}$.

For this purpose, we first illustrate how to relate infinite traces (i.e., pairs of vectors of fps) and vectors of fls. We say that a trace $\left(\overrightarrow{\alpha}, \overrightarrow{\beta}\right) \in \mathsf{k}[[x]]^n \times \mathsf{k}[[x]]^m$ *generates* $\left(\overrightarrow{\sigma}, \overrightarrow{\tau}\right) \in \mathsf{k}((x))^n \times \mathsf{k}((x))^m$ if there exist an *instant* $z \in \mathbb{Z}$ such that

(i) $\alpha_j(i) = \sigma_j(i + z)$ and $\beta_h(i) = \tau_h(i + z)$ for all $i \in \mathbb{N}$ and k, j with $1 \leq j \leq n$, $1 \leq h \leq m$;

(ii) z is smaller or equal than any degree of $\sigma_1 \ldots \sigma_n, \tau_1 \ldots \tau_m$.

To see the intuition behind this notion, recall from Section 3 that, whereas fps give a way of encoding streams, fls encode streams "with a (finite) past". If we see it as a translation from $(\overrightarrow{\alpha}, \overrightarrow{\beta})$ to $(\overrightarrow{\sigma}, \overrightarrow{\tau})$, our correspondence takes the fps in $(\overrightarrow{\alpha}, \overrightarrow{\beta})$ and fix for them all a common "present" moment. For example, the trace $(k_0 k_1 k_2 \ldots, l_0 l_1 l_2 \ldots) \in \mathsf{k}[[x]] \times \mathsf{k}[[x]]$ generates infinitely many pairs of fls, among which we have:

$$(\ldots 0\underline{0} k_1 k_2 k_3 \ldots, \ldots 0\underline{0} l_1 l_2 l_3 \ldots)$$
$$(\ldots 00\underline{k_1} k_2 k_3 \ldots, \ldots 00\underline{l_1} l_2 l_3 \ldots)$$
$$(\ldots 00\overline{k_1}\,\underline{k_2} k_3 \ldots, \ldots 00\overline{l_1}\,\underline{l_2} l_3 \ldots)$$

[1] See [7, §7.2] for a more detailed account of this transformation in a denotational context.

with choice of present moment $(0,0)$, (k_1,l_1) and (k_2,l_2) respectively. The instant $z \in \mathbb{Z}$ will be 1 for the first, 0 for the second and -1 for the third pair above.

Conversely, we can start from all the fls in $(\overrightarrow{\sigma},\overrightarrow{\tau})$ and forget about their present moment to obtain streams. The requirement that the instant z is chosen not bigger than any degree implies that only 0s are removed in the process, that is, there is no information loss. For instance, $(\dots 00k_1\underline{k_2}k_3\dots,\ \dots 00\underline{l_1}l_2l_3\dots) \in \mathsf{k}((x))\times \mathsf{k}((x))$ is generated by infinitely many traces, including:

$$\begin{array}{c}(k_1k_2k_3\dots,\ 0l_1l_2l_3\dots)\\(0k_1k_2k_3\dots,\ 00l_1l_2l_3\dots)\\(00k_1k_2k_3\dots,\ 000l_1l_2l_3\dots).\end{array}$$

The instant z is chosen to be -1 for the first, -2 for the second and -3 for the third pair above.

To complete the picture, we relate finite and infinite traces. A trace $(\overrightarrow{\alpha},\overrightarrow{\beta}) \in \mathsf{k}[x]^n \times \mathsf{k}[x]^m$ of length z is a *prefix* of an infinite trace $(\overrightarrow{\gamma},\overrightarrow{\delta}) \in \mathsf{k}[[x]]^n \times \mathsf{k}[[x]]^m$ iff $\alpha_j(i) = \gamma_j(i)$ and $\beta_h(i) = \delta_h(i)$ for all $0 \le i \le z$, $1 \le j \le n$ and $1 \le h \le m$.

We are now ready to define our correspondence between arrows of $\mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$ and arrows of $\mathsf{Rel}_{\mathsf{k}((x))}$.

Definition 4. Let (f,g) be an arrow of $\mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}$. We define $\mathcal{F}(f,g)$ as the following arrow in $\mathsf{Rel}_{\mathsf{k}((x))}[n,m]$:

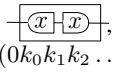
$$\left\{\left(\overrightarrow{\sigma},\overrightarrow{\tau}\right) \;\middle|\; \begin{array}{l}\text{there exist a trace } \left(\overrightarrow{\alpha},\overrightarrow{\beta}\right) \in g\\ \text{generating } \left(\overrightarrow{\sigma},\overrightarrow{\tau}\right)\end{array}\right\}$$

In the converse direction, given an arrow $S \in \mathsf{Rel}_{\mathsf{k}((x))}[n,m]$, we define $\mathcal{U}(S)$ as the pair $(f,g) \in \mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}[n,m]$ where $g \in \mathsf{Rel}_{\mathsf{k}[[x]]}[n,m]$ is given as

$$\left\{\left(\overrightarrow{\alpha},\overrightarrow{\beta}\right) \;\middle|\; \begin{array}{l}\text{there exist a pair } \left(\overrightarrow{\sigma},\overrightarrow{\tau}\right) \in S\\ \text{generated by } \left(\overrightarrow{\alpha},\overrightarrow{\beta}\right)\end{array}\right\}$$

and $f \in \mathsf{Rel}_{\mathsf{k}[x]}[n,m]$ is the set of all prefixes of the traces in g.

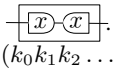
Intuitively, the action of $\mathcal{F}$ on (f,g) is to forget the first component and generate all the vectors of fls generated by vectors of fps in g. Instead we can describe $\mathcal{U}$ as abstracting away the choice of the present for all fls and represent them as fps. This gives the second element in the target pair (f,g): the first is irrelevant, since it only consists of the prefixes of traces in g (in particular, we do not generate any deadlock trace). To see how $\mathcal{F}$ and $\mathcal{U}$ work more precisely, we shall consider the following example.

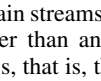
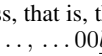
Example 3. Recall that, for $c = $  , the set $it(c)$ consists of all infinite traces of the form $(0k_0k_1k_2\dots, 0k_0k_1k_2\dots)$, that is, c behaves as the identity after one initialisation step. Since this circuit is deadlock free, the set $ft(c)$ instead contains all and only those finite traces which are prefixes of some infinite trace in $it(c)$.

One can check that $\mathcal{F}\langle c\rangle$ is the identity relation $\{(\sigma,\sigma) \mid \sigma \in \mathsf{k}((x))\}$ (which is actually equal to $[\![c]\!]$). For instance, the pair of fls

$$(\dots 00\underline{k_0}k_1k_2k_3\dots,\ \dots 00\underline{k_0}k_1k_2k_3\dots) \tag{12}$$

is generated by $(0k_0k_1k_2\dots, 0k_0k_1k_2\dots)$. If we then apply $\mathcal{U}$ to $\mathcal{F}\langle c\rangle$, we obtain a pair $(f,g) \in \mathsf{Rel}_{\mathsf{k}[x]} \times \mathsf{Rel}_{\mathsf{k}[[x]]}[1,1]$ where f coincides with the original $ft(c)$, while g is strictly larger than $it(c)$. Indeed $(k_0k_1k_2\dots, k_0k_1k_2\dots) \notin it(c)$ but it belongs to g since it generates the pair (12).

We now focus on circuit $d = $  . The set $it(d)$ consists of all infinite traces of the form $(k_0k_1k_2\dots, k_0k_1k_2\dots)$ and the set $ft(d)$ consists of either prefixes of $it(d)$ or traces leading to deadlocks having the form $(k_0k_1\dots k_ul, k_0k_1\dots k_ul')$ with $l \neq l'$. It is easy to check that these traces are lost when applying $\mathcal{U}\mathcal{F}$ to $\langle d\rangle$, while no infinite trace is added or removed.

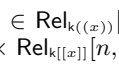
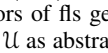
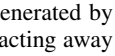
The example above suggests that the composite mapping $\mathcal{F}\mathcal{U}$ *enlarges* the set of observable behaviors for circuits with initialisation (e.g. ) and, dually, *restricts* it for circuits with deadlocks (e.g. ). The next statement illustrates the extent of these observations.

Theorem 4. Let c be a circuit of Circ. Then the following holds:

(a) $\mathcal{F}\langle c\rangle = [\![c]\!]$.
(b) If c is deadlock free, then $\langle c\rangle \subseteq \mathcal{F}\mathcal{U}[\![c]\!]$. [2]
(c) If c is initialisation free, then $\langle c\rangle \supseteq \mathcal{F}\mathcal{U}[\![c]\!]$.

Statement (a) above is instrumental in showing full abstraction (Corollary 2), but is also of independent interest. Indeed it allows to immediately derive that

Corollary 1. For any two circuits $c, d \in$ Circ, $[\![c]\!] = [\![d]\!]$ if and only if $\mathcal{F}\langle c\rangle = \mathcal{F}\langle d\rangle$.

In some sense, Corollary 1 tells us under which conditions an external observer cannot distinguish circuits that have the same denotation. This is the case whenever $\mathcal{F}\langle c\rangle = \mathcal{F}\langle d\rangle$, that is, the observation of c and d can be only made "up-to $\mathcal{F}$". Intuitively, this amounts to imposing the following two conditions, stemming from the definition of $\mathcal{F}$. First, we prevent the observation of finite behavior — because $\mathcal{F}$ disregards $ft(c)$ and $ft(d)$. This means that we cannot detect deadlock and, for instance,  and  become indistinguishable. Second, we prevent an external agent from choosing *when* to begin the observation. For instance, take $c = $. By observing the pair $(\dots 00k_0\underline{k_1}k_2\dots, \dots 00k_0\underline{k_1}k_2\dots)$ in $\mathcal{F}\langle c\rangle$, on principle we are not able to judge whether it has been generated by a trace $(k_0k_1k_2\dots, k_0k_1k_2\dots)$ or $(0k_0k_1k_2\dots, 0k_0k_1k_2\dots)$ in $\langle c\rangle$: the definition of $\mathcal{F}$ allows for both (and infinitely many other) options. Since our view is restricted to $\mathcal{F}\langle c\rangle$, we cannot tell if the observation of the actual stream starts with 0 or k_0. Therefore, from that viewpoint  and  are indistinguishable.

In general, when observations can be made without the restrictions of $\mathcal{F}$, one *can* distinguish amongst circuits that are denotationally equivalent, as explained in Example 3. Statements (b) and (c) in Theorem 4 allow us to derive that observations and denotations do coincide for the class of well-behaved circuits that do not suffer from deadlocks and initialisation steps.

Corollary 2 (Full Abstraction)**.** For any two circuits c and d of Circ that are deadlock and initialisation free,

$$[\![c]\!] = [\![d]\!] \text{ if and only if } \langle c\rangle = \langle d\rangle.$$

5. Realisability

In the previous section, we have seen two different canonical forms for circuits of Circ: the span form, preventing deadlocks, and the cospan form, avoiding initialisation steps. We now tackle the question of transforming, within the equational theory of $\mathbb{IH}$, any circuit c of Circ into one d featuring both properties. This is appealing because, by Corollary 2, d properly *realises* the denoted behaviour $[\![c]\!]$. To this aim, we first observe that our desiderata are fulfilled for the class SF, introduced in Section 2:

Proposition 3. Every c in SF is deadlock and initialisation free.

[2] The inclusion is meant to be component-wise, i.e. $it(c) \subseteq g$ and $ft(c) \subseteq f$ where $(ft(c), it(c)) = \langle c\rangle$ and $(f,g) = \mathcal{F}\mathcal{U}[\![c]\!]$.

The intuition behind Proposition 3 is that the rules for inductively constructing circuits of SF do not allow for the conflicting design originating the phenomena of initialisation and deadlock. In particular, adding feedbacks to a circuit of SF preserves initialisation and deadlock freedom.

In order to achieve our goal, we will introduce a canonical form for circuits in Circ which is based on SF. Roughly speaking, such form consists of signal flow graphs where "the wires have been jumbled up". We make this intuition precise in the sequel.

First, for each $n, m \in \mathbb{N}$, we take circuits $\eta_n \colon n \to 1+1+n$ and $\epsilon_m \colon 1+1+m \to m$ as illustrated below.

$\eta_n :=$  $\qquad \epsilon_m :=$ 

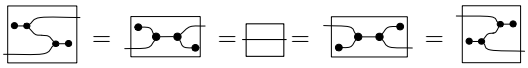
Then, we define the families of operators $\mathsf{L}_{n,m} \colon \mathsf{Circ}[n+1, m] \to \mathsf{Circ}[n, 1+m]$ and $\mathsf{R}_{n,m} \colon \mathsf{Circ}[n, 1+m] \to \mathsf{Circ}[1+n, m]$ as follows: for any circuit $c \in \mathsf{Circ}[n+1, m]$,

$$\mathsf{L}_{n,m}(c) = \eta_n \,;(id_1 \oplus c) \quad \left(\begin{array}{c} \\ n \boxed{c} m \end{array} \right)$$

and, for any circuit $d \in \mathsf{Circ}[n, m+1]$

$$\mathsf{R}_{n,m}(d) = (id_1 \oplus d) \,;\epsilon_m. \quad \left(\begin{array}{c} \\ n \boxed{d} m \end{array} \right)$$

Remark 4. When defined on $\mathbb{IH}$, $\mathsf{L}_{n,m}$ and $\mathsf{R}_{n,m}$ enjoy some interesting properties. Let $1 + - \colon \mathbb{IH} \to \mathbb{IH}$ be the functor acting on objects as $k \mapsto 1+k$ and on arrows as $f \mapsto id_1 \oplus f$. This functor is self-adjoint: the unit and the counit are the η_n and ϵ_m defined as above. The fact that $\mathbb{IH}$ is a SMC implies naturality of η and ϵ. They satisfy the triangle inequalities because of the following "snake" lemma, provable in $\mathbb{IH}$ using (I2), (A5) and (A5)*:

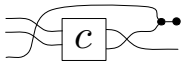
 (13)

The canonical isomorphisms between hom-sets, induced by the adjunction, are $\mathsf{L}_{n,m} \colon \mathbb{IH}[n+1, m] \to \mathbb{IH}[n, 1+m]$ and $\mathsf{R}_{n,m} \colon \mathbb{IH}[n, 1+m] \to \mathbb{IH}[1+n, m]$ defined as above. We can understand $\mathsf{L}_{n,m}$ intuitively as "rewiring" the connection to the first port on the left to the right of the circuit. The fact that $\mathsf{L}_{n,m}$ and $\mathsf{R}_{n,m}$ are isomorphisms means that no information is lost – all such circuits can be "rewired" back to their original form.

Definition 5. Any $c_2 \in \mathsf{Circ}[n_2, m_2]$ is a *rewiring* of $c_1 \in \mathsf{Circ}[n_1, m_1]$ when c_2 can be obtained from c_1 by a combination of the following operations:

- application of $\mathsf{L}_{n,m}$, for some n and m,
- application of $\mathsf{R}_{n,m}$, for some n and m,
- post-composition with a permutation,
- pre-composition with a permutation.

Permutations are needed to rewire any port on the boundaries. For instance, they allow to rewire the second port on the right as the third on the left in the circuit $c \colon 2 \to 2$ below:



For our purposes, it is of importance to observe that

Lemma 1. Rewiring preserves deadlock and initialisation freedom.

The remainder of this section will be devoted to showing that we can put circuits of Circ into the desired shape.

Theorem 5. Every circuit in Circ is equal in $\mathbb{IH}$ to the rewiring of some circuit in SF.

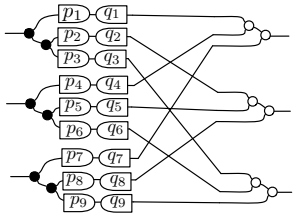
By Theorem 5, Proposition 3 and Lemma 1 we achieve the realisability result claimed at the beginning of this section.

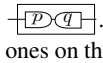
Corollary 3 (Realisability)**.** Every circuit c of Circ is equal in $\mathbb{IH}$ to some circuit d of Circ that is deadlock and initialisation free.

5.1 Proof of Theorem 5

In order to prove Theorem 5, we first need some preliminaries on matrices and their circuit representations. Indeed, there is a natural way of representing matrices over the field $\mathsf{k}(x)$ of fractions of polynomials as circuits of Circ. To illustrate it, consider the following example:

$$\begin{pmatrix} p_1/q_1 & p_4/q_4 & p_7/q_7 \\ p_2/q_2 & p_5/q_5 & p_8/q_8 \\ p_3/q_3 & p_6/q_6 & p_9/q_9 \end{pmatrix}$$



The circuit c on the right encodes the matrix M on the left in the following way: for each boundary of c, we assume a top-bottom enumerations of the ports, starting from 1. Then the entry $M_{j,i}$ (column j, row i) has value $\frac{p}{q} \in \mathsf{k}(x)$ if and only if, reading the circuit from the left to the right, one finds a path connecting the j^{th} port on the left to the i^{th} port on the right passing through a circuit  . Intuitively, the ports on the left represent columns, the ones on the right rows, and the links between them carry the values in the matrix. One can verify the correspondence between c and M formally: $[\![c]\!]$ is the relation $\{(\overrightarrow{\sigma}, M \cdot \overrightarrow{\sigma}) \mid \overrightarrow{\sigma} \in \mathsf{k}((x))^3\}$.

It will be convenient to work with matrices (and the corresponding circuits) of a particular shape. We say that a matrix over $\mathsf{k}(x)$ is in *rational form* if all its entries are in fact rationals (in $\mathsf{k}\langle x \rangle$) and:

1. for each non-zero row, there is a pivot entry with value 1.
2. in the column of a pivot, the pivot is the only non-zero entry.

An example is displayed below where $r_1, r_2, r_3 \in \mathsf{k}\langle x \rangle$.

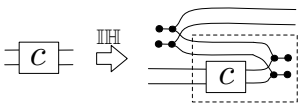
$$\begin{pmatrix} r_1 & 0 & 1 & 0 \\ r_2 & 1 & 0 & 0 \\ r_3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The following lemma is the final ingredient for the proof of Theorem 5—its proof is an easy exercise in linear algebra.

Lemma 2. Every $\mathsf{k}(x)$ matrix is row equivalent to one in rational form.

Proof of Theorem 5. Let us fix a circuit $c \in \mathsf{Circ}[n, m]$. In the following, we will sketch a recipe, using the equational theory of $\mathbb{IH}$, with which c is transformed into the rewiring of a circuit in SF. To improve readability, we shall draw any circuit as if both n and m were 2. It should be fairly clear how our argument generalizes.

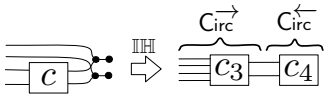
(i) First we transform c into the circuit c_1 on the right: the two are equal in $\mathbb{IH}$ by virtue of (13).



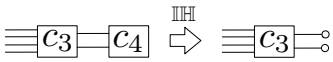
Let us call c_2 the circuit from $n+m$ to 0 delimited by a dotted square in the picture above. Since c_0 is obtained by rewiring on

c_2, it should be clear that, if c_2 can be rearranged as the rewiring of a circuit in SF, then this property also holds for c_0. Therefore, in the sequel we shift our focus to c_2.

(ii) Proposition 1 allows us to rewrite c_2 in cospan form, as the composition along a middle boundary z of circuits c_3 and c_4 below, while preserving equality in $\mathbb{IH}$. By definition of cospan form, c_3 is an arrow of $\mathsf{C}\overrightarrow{\mathsf{irc}}$, while c_4 is an arrow of $\mathsf{C}\overleftarrow{\mathsf{irc}}$. For the sake of readability, we will draw z as if it were 2.

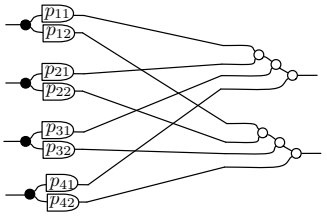


(iii) Since we are reasoning in $\mathbb{IH}$, in particular all the equations of $\mathbb{HA}^{op}$ hold. Now, 0 is both the initial and the terminal object in $\mathsf{Mat}\,\mathsf{k}[x]$; because $\mathbb{HA}^{op} \cong \mathsf{Mat}\,\mathsf{k}[x]^{op}$, this means that there is exactly one circuit of $\mathsf{C}\overleftarrow{\mathsf{irc}}$, up to equivalence in $\mathbb{HA}^{op}$, from z to 0. It follows that c_4 and the z-tensor of  (a circuit that we call c_5) are equal in $\mathbb{HA}^{op}$ — and thus in $\mathbb{IH}$. We can thus write $c_3\,;c_4$ as $c_3\,;c_5$:

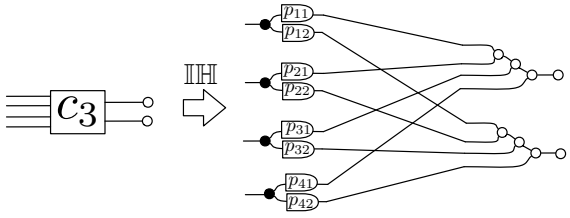


(iv) Since c_3 is in $\mathsf{C}\overrightarrow{\mathsf{irc}}$ we can use the results about $\mathbb{HA}$ to reason about it. In particular, c_3 can be mapped into an $z \times (m+n)$ matrix of polynomials M, because $\mathbb{HA} \cong \mathsf{Mat}\,\mathsf{k}[x]$. As we remarked at the beginning of this section, there is a canonical way of representing M as a circuit c_6 of Circ:

$$\begin{pmatrix} p_{11} & p_{21} & p_{31} & p_{41} \\ p_{12} & p_{22} & p_{32} & p_{42} \end{pmatrix}$$

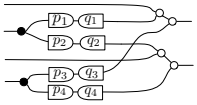


Since c_3 corresponds to M along the isomorphism $\mathbb{HA} \cong \mathsf{Mat}\,\mathsf{k}[x]$, it follows that $[\![c_3]\!]$ is $\{(\sigma, M\cdot\sigma \mid \sigma \in \mathsf{k}((x))^n\}$. Therefore $[\![c_3]\!] = [\![c_6]\!]$ meaning by Theorem 1 that $c_3 \overset{\mathbb{IH}}{=} c_6$. It follows that we can rewrite our circuit $c_3\,;c_5$ as $c_6\,;c_5$:



(v) Using Lemma 2, we can then transform M into a matrix M^* in rational form (for instance, the one on the left below). Since M^* is a matrix over $\mathsf{k}(x)$, we have a canonical circuit c_7 of $\mathbb{IH}$ (on the right) representing it.

$$\begin{pmatrix} 1 & p_1/q_1 & 0 & p_3/q_3 \\ 0 & p_2/q_2 & 1 & p_4/q_4 \end{pmatrix}$$

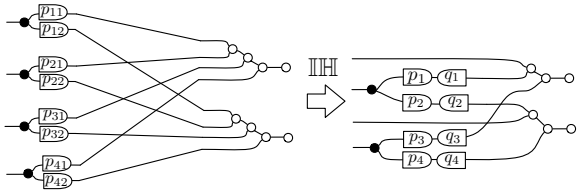


By definition of rational form, each non-zero row R in M^* is associated with a pivot column C with the only non-zero value 1 at the intersection of R and C. In order to graphically represent such property in c_7, we supposed the following choice of pivots: the first and the third column for the first and second row respectively. Observe that an entry with value 0 corresponds to the circuit  , which in $\mathbb{IH}$ is equal to : therefore we could avoid drawing the corresponding link in the circuit c_7.

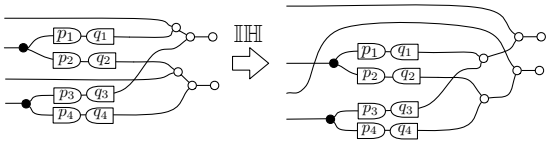
We now claim that $c_6\,;c_5 \overset{\mathbb{IH}}{=} c_7\,;c_5$. By Theorem 1, to check this it suffices to show that $[\![c_6\,;c_5]\!] = [\![c_7\,;c_5]\!]$, that is:

$$\{(\sigma, M\cdot\sigma \mid M\cdot\sigma = \overrightarrow{0}\} = \{(\sigma, M^*\cdot\sigma) \mid M^*\cdot\sigma = \overrightarrow{0}\}.$$

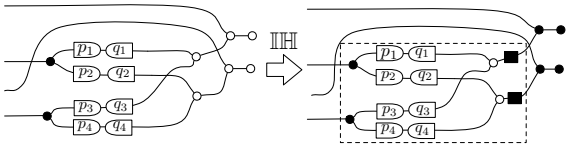
This is true because the two relations above describe the kernel of M and M^* respectively, and M^* is row-equivalent to M. It follows that we can rewrite $c_6\,;c_5$ as $c_7\,;c_5$, while preserving equality in $\mathbb{IH}$:

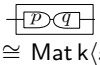


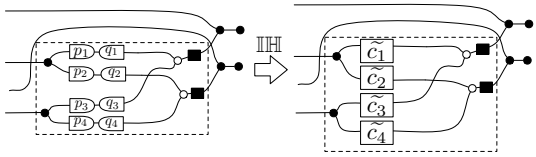
(vi) We now focus on circuit $c_7\,;c_5$. Our next step is to use associativity and commutativity of  to make one of the two legs of each component  be always attached to the pivot-wire of the corresponding row. Also, we use the axioms of SMCs and naturality of the symmetry  to push the pivot-wires towards the top of the circuit, as follows:



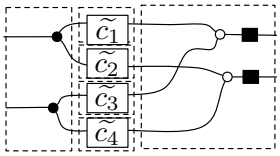
(vii) We can now remove the components of shape  by turning them into rewiring structure. This can be done by simply applying axiom (I6) of $\mathbb{IH}$:



(viii) Let us call c_8 the rightmost circuit above: it is a rewiring of the circuit inscribed into the dotted square, which we call c_9. Since c_7 was constructed starting by a matrix in rational form, for all the components  in c_8, $\frac{p}{q}$ is a rational. Thus, using the fact that $\mathbb{SF} \cong \mathsf{Mat}\,\mathsf{k}\langle x\rangle$, we can rewrite in $\mathbb{IH}$ each such component as a circuit $\tilde{c}$ in SF:



Now, observe that c_9 can be seen as the composition (via $\oplus$ and ;) of circuits that are all in SF.

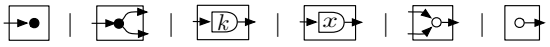


It follows that c_9 is also in SF and thus c_8 is the rewiring of a circuit in SF. Since c_8 was obtained by c_2 by only using rewriting steps allowed by the equational theory of $\mathbb{IH}$, the statement of the theorem follows. □

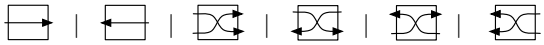
6. Directing the Flow

In the classical presentation of signal flow graphs (see e.g. [23]), wires are directed, signifying the direction of signal flow. Throughout the previous sections, we have been referring to flow direction only on an intuitive level. We now introduce directionality explicitly, claiming that it can be really treated as a *derivative* notion of our theory of circuits. We then present some applications and examples supporting our statement.

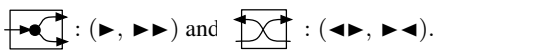
In order to model classical signal flow graphs we first need to introduce an alternative syntax, which we call the *directed* signal flow calculus. We will need components that resemble those of $\mathsf{C}\overrightarrow{\mathsf{irc}}$, but which are explicitly oriented from left to right.

$e ::=$ 

We also require some "pure" wiring: since signal flow is explicit, we include two versions of the identity wire and four of the twist:

$w ::=$ 

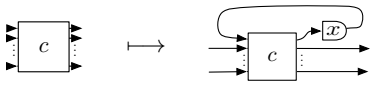
These basic components above are given a sorting (u, v) where $u, v \in \{\blacktriangleleft, \blacktriangleright\}^*$; for instance:



Classical signal flow graphs are obtained by composing components e and w using the operations $;$ and $\oplus$, for which we reuse the sorting rules of Fig. 1, together with guarded feedback operations $\mathsf{Tr}^{\blacktriangleright}(\cdot)$ that take a circuit of sort $(\blacktriangleright^{1+m}, \blacktriangleright^{1+n})$ and yield a circuit of sort $(\blacktriangleright^m, \blacktriangleright^n)$. The associated sorting rule is thus:

$$\frac{c : (\blacktriangleright^{1+n}, \blacktriangleright^{1+m})}{\mathsf{Tr}^{\blacktriangleright}(c) : (\blacktriangleright^n, \blacktriangleright^m)}$$

This is represented graphically as follows:



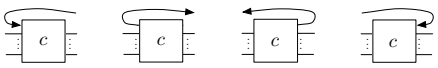
The syntax for directed signal flow graphs is thus:

$$sf ::= e \mid w \mid sf\,;sf \mid sf \oplus sf \mid \mathsf{Tr}^{\blacktriangleright}(sf)$$

Finally, we include top-level operations reminiscent of the rewiring in §5: $L^{\blacktriangleright}$, $L^{\blacktriangleleft}$, $R^{\blacktriangleright}$ and $R^{\blacktriangleleft}$, with sorting rules:

$$\frac{c : (\blacktriangleright u, v)}{L^{\blacktriangleright}(c) : (u, \blacktriangleleft v)} \quad \frac{c : (\blacktriangleleft u, v)}{L^{\blacktriangleleft}(c) : (u, \blacktriangleright v)} \quad \frac{c : (u, \blacktriangleright v)}{R^{\blacktriangleright}(c) : (\blacktriangleleft u, v)} \quad \frac{c : (u, \blacktriangleleft v)}{R^{\blacktriangleleft}(c) : (\blacktriangleright u, v)}$$

In the graphical rendering below we leave out the arrowheads on wires where direction is arbitrary:

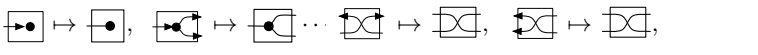


Circuits of the directed signal flow calculus are thus specified by following grammar:

$$d ::= sf \mid L^{\blacktriangleright} d \mid L^{\blacktriangleleft} d \mid R^{\blacktriangleright} d \mid R^{\blacktriangleleft} d \mid d\,;w \mid w\,;d$$

Note that the composition at the top level is restricted to disallow the introduction of unguarded feedback.

Rather than defining the operational semantics directly, we can obtain the expected behaviour by first translating directed terms to the signal flow calculus. Intuitively, the inductively defined translation $\mathcal{E}$ "erases directions" from the wires:



$$sf_1\,;sd_2 \mapsto \mathcal{E}(sf_1)\,;\mathcal{E}(sf_2), \quad sf_1 \oplus sf_2 \mapsto \mathcal{E}(sd_1) \oplus \mathcal{E}(sf_2),$$
$$\mathsf{Tr}^{\blacktriangleright}(sf) \mapsto \mathsf{Tr}(\mathcal{E}(sf)), \quad \mathsf{L}^{\star}(d) \mapsto \mathsf{L}(\mathcal{E}(d)) \quad \mathsf{R}^{\star}(d) \mapsto \mathsf{R}(\mathcal{E}(d)).$$

where $\star \in \{\blacktriangleleft, \blacktriangleright\}$ and Tr, L and R are defined as in §2 and §5.

A key observation is that directed sort discipline prevents us from writing problematic circuits where signal flow is incompatible, like in the examples in §4. In fact, using Proposition 3 and Lemma 1 we get:

Proposition 4. For any circuit d of the directed signal flow calculus, $\mathcal{E}(d)$ is deadlock and initialisation free.

Moreover, this syntactic restriction does not affect the expressiveness since, thanks to Theorem 5, rewirings of signal flow graphs denote all the possible behaviours. Thus, informally speaking, all circuits in Circ can be directed (modulo the theory of $\mathbb{IH}$).

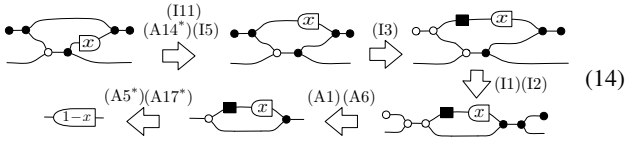
Proposition 5. For any circuit c of Circ, there exists a directed circuit d such that $\mathcal{E}(d) \overset{\mathbb{IH}}{=} c$

Propositions 4 and 5 have two interesting consequences. First, Proposition 4 and the full-abstraction result mean that we can use the equational theory of $\mathbb{IH}$ to safely reason about classical signal flow graphs and extensions—indeed, all the circuits in the directed signal flow calculus. Roughly speaking, the procedure is: forget the directions and then use $\overset{\mathbb{IH}}{=}$. This confirms the intuition that, like for electrical circuits, also for signal flow graphs directionality is *not* a primitive notion as originally advocated in [23].

Second, Proposition 4, Proposition 5 and full-abstraction tell us that the denotational semantics of any circuit of the signal flow calculus can be properly realised by some directed circuit. We can therefore use the "more liberal" signal flow calculus to specify circuits and the "more restrictive" directed calculus to implement them. One can then check that an implementation d adheres to a specification c by mean of the graphical reasoning supported by $\mathbb{IH}$. Indeed $\mathcal{E}(d) \overset{\mathbb{IH}}{=} c$, means that d implements, without deadlocks or initialisation, the behaviour denoted by c. Note that while an implementation is a directed circuit—typically featuring feedbacks—we are being deliberately vague about what kind of circuit in Circ constitutes a specification: in examples that we consider these are typically generating functions that can be obtained in a standard way (see e.g. [29]) from recurrence formulas. We illustrate these ideas with the aid of the simple example below.

Example 4. Consider the circuits displayed below. The leftmost serves as specification ($\frac{1}{1-x}$) and the rightmost, a directed circuit, as its implementation.

To prove that the implementation realises the specification, we first throw away all the directions from the wires and then we proceed with a graphical derivation in $\mathbb{IH}$:

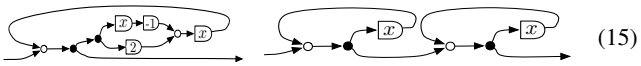


(14)

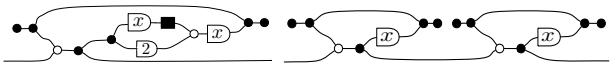
We annotated the key axioms of $\mathbb{IH}$ justifying each derivation step. Note that the first and the second to last circuit, that we have just

proved equivalent, are c_2 and c_1 from Example 1 (modulo the notation  adopted for since Section 3).

A similar procedure can be used to check the observational equivalence of directed signal flow graphs. For instance, take:

 (15)

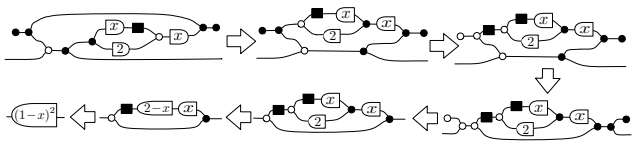
First, we forget the direction of the flow and we obtain the circuits c_3 and c_4 depicted below, on the left and on the right.

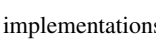


Then, by virtue of Proposition 4 and full abstraction, we can safely use $\overset{\mathbb{IH}}{=}$ to check $\langle c_3 \rangle = \langle c_4 \rangle$. Observe that c_3 is like in Example 1 and c_4 is just the sequential composition $c_2 \,;\, c_2$. We can thus reuse (14) to see that

$$c_4 = c_2 \,;\, c_2 \overset{\mathbb{IH}}{=} \boxed{1-x} \,;\, \boxed{1-x} \overset{\mathbb{IH}}{=} \boxed{(1-x)^2}$$

To conclude, we only have to check that c_3 is equal in $\mathbb{IH}$ to the rightmost circuit above. This is shown as follows, along the same lines of derivation (14):



The circuits in (15) can also be thought of as two different implementations of  . Indeed, $\frac{1}{(1-x)^2}$ is the generating function of the sequence $1, 2, 3, 4, \ldots$.

7. Conclusions

The network theoretic approach combines algebra and topology–the circuits of the theory that we presented have an algebraic nature, as demonstrated by the axiomatisations, as well as a topological nature, when viewed as string diagrams. Our contribution adds an operational understanding to the previously discovered denotational insights. Throughout the paper we have tried to illustrate the fruitful interplay between algebra, topology, the operational and denotational approaches.

Although our attention in this work was restricted to signal flow graphs, the same methodology could be beneficial in other areas where diagrammatic notation is employed: in addition to the examples we mentioned in the introduction there are Kahn process networks, Bayesian networks and automata, amongst many others. Typically, such diagrammatic formalisms are translated to more traditional mathematics, but seldom reasoned about directly. The broad picture of the work in this paper is a deep connection between a denotational view and a fully-fledged operational approach that is intimately related to the hallmark of network theory: the interplay between algebra and topology. Our vision is close to that advocated by Abramsky for concurrency theory [1]: we believe that this approach will eventually lead to less a specialised, fragmented and sometimes overly syntax-focussed landscape.

Acknowledgements We thank the anonymous referees for the fruitful discussion. The first and the third author acknowledge support from the ANR project 12IS02001 PACE.

References

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

[2] J. C. Baez. Network theory. `http://math.ucr.edu/home/baez/networks/`, 2014.

[3] J. C. Baez and J. Erbele. Categories in control. *CoRR*, abs/1405.6881, 2014. `http://arxiv.org/abs/1405.6881`.

[4] H. Basold, M. Bonsangue, H. H. Hansen, and J. Rutten. (co)algebraic characterizations of signal flow graphs. In *To appear in LNCS*, 2014.

[5] F. Bonchi, P. Sobociński, and F. Zanasi. A categorical semantics of signal flow graphs. In *CONCUR*, 2014.

[6] F. Bonchi, P. Sobociński, and F. Zanasi. Interacting bialgebras are Frobenius. In *FoSSaCS '14*. Springer, 2014.

[7] F. Bonchi, P. Sobociński, and F. Zanasi. Interacting Hopf algebras. *CoRR*, abs/1403.7048, 2014. `http://arxiv.org/abs/1403.7048`.

[8] R. Bruni, U. Montanari, G. Plotkin, and D. Terreni. On hierarchical graphs: reconciling bigraphs, gs-monoidal theories and gs-graphs.

[9] R. Bruni, I. Lanese, and U. Montanari. A basic algebra of stateless connectors. *Theor Comput Sci*, 366:98–120, 2006.

[10] A. Carboni and R. F. C. Walters. Cartesian bicategories I. *J Pure Appl Algebra*, 49:11–32, 1987.

[11] B. Coecke and R. Duncan. Interacting quantum observables. In *ICALP'08*, pages 298–310, 2008.

[12] B. Coecke, R. Duncan, A. Kissinger, and Q. Wang. Strong complementarity and non-locality in categorical quantum mechanics. In *LiCS'12*, pages 245–254, 2012.

[13] M. P. Fiore and M. D. Campos. The algebra of directed acyclic graphs. In *Abramsky Festschrift*, volume 7860 of *LNCS*, 2013.

[14] B. Fong. A compositional approach to control theory. PhD Transfer Report, 2013.

[15] D. R. Ghica. Diagrammatic reasoning for delay-insensitive asynchronous circuits. In *Abramsky Festschrift*, pages 52–68, 2013.

[16] A. Joyal and R. Street. The geometry of tensor calculus, I. *Adv. Math.*, 88:55–112, 1991.

[17] P. Katis, N. Sabadini, and R. F. C. Walters. Span(Graph): an algebra of transition systems. In *AMAST '97*, pages 322–336. Springer, 1997.

[18] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *J. Pure Appl. Algebra*, 19:193–213, 1980.

[19] S. Lack. Composing PROPs. *Theor App Categories*, 13(9):147–163, 2004.

[20] Y. Lafont. Towards an algebraic theory of boolean circuits. *J Pure Appl Alg*, 184:257–310, 2003.

[21] B. Lahti. *Signal Processing and Linear Systems*. Oxford University Press, 1998.

[22] S. Mac Lane. Categorical algebra. *B Am Math Soc*, 71:40–106, 1965.

[23] S. J. Mason. *Feedback Theory: I. Some Properties of Signal Flow Graphs*. MIT Research Laboratory of Electronics, 1953.

[24] D. Pavlovic. Monoidal computer i: Basic computability by string diagrams. *Inf. Comput.*, 226:94–116, 2013.

[25] D. Pavlovic. Monoidal computer ii: Normal complexity by string diagrams. *CoRR*, abs/1402.5687, 2014.

[26] J. J. M. M. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *Theor. Comput. Sci.*, 343(3):443–481, 2005.

[27] P. Selinger. A survey of graphical languages for monoidal categories. arXiv:0908.3347v1 [math.CT], 2009.

[28] P. Sobociński. Nets, relations and linking diagrams. In *CALCO '13*, 2013.

[29] H. Wilf. *Generatingfunctionology*. A. K. Peters, 3rd edition, 2006.

[30] J. C. Willems. The behavioural approach to open and interconnected systems. *IEEE Contr. Syst. Mag.*, 27:46–99, 2007.

[31] W. J. Zeng and J. Vicary. Abstract structure of unitary oracles for quantum algorithms. *CoRR*, abs/1406.1278, 2014.