UNIVERSITY OF CALIFORNIA
RIVERSIDE

Categories in Control: Applied PROPs

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Mathematics

by

Jason Michael Erbele

December 2016

Dissertation Committee:

Dr. John C. Baez, Chairperson
Dr. Julie Bergner
Dr. Kevin Costello

The Dissertation of Jason Michael Erbele is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

Large parts of Chapters 1 and 3 of the current work appeared in 2015 in *Theories and Applications of Categories*, Volume 30. This prior incarnation of *Categories in control* has been expanded to the present corpus. I am grateful for the direction and supervision of John Baez, whose clear exposition and advice have been incredibly useful. I could not have completed this dissertation without his guidance, prodding, and helpful meddling. Also valuable were the many conversations with Brendan Fong, which helped to crystallize several ideas.

Most of all, I owe an enormous debt of gratitude to the late Dr. Gene Scott. The scope of that debt is too large to fit in this section. The least of which, his leadership and tenacity inspired me to pursue higher education and to persevere when the path looked impossible.

To the young at heart,

to the curious in mind,

to the kindred soul.

ABSTRACT OF THE DISSERTATION

Categories in Control: Applied PROPs

by

Jason Michael Erbele

Doctor of Philosophy, Graduate Program in Mathematics
University of California, Riverside, December 2016
Dr. John C. Baez, Chairperson

Control theory uses 'signal-flow diagrams' to describe processes where real-valued functions of time are added, multiplied by scalars, differentiated and integrated, duplicated and deleted. These diagrams can be seen as string diagrams for the PROP $\mathtt{FinRel}_k$, the strict version of the category of finite-dimensional vector spaces over the field of rational functions $k = \mathbb{R}(s)$ and linear relations, where the variable s acts as differentiation and the monoidal structure is direct sum rather than the usual tensor product of vector spaces. Control processes are also described by controllability and observability—whether the input can drive the process to any state, and whether any state can be determined from later outputs. For any field k we give a presentation of $\mathtt{FinRel}_k$ in terms of generators of the free PROP of signal-flow diagrams together with the equations that give $\mathtt{FinRel}_k$ its structure. The 'cap' and 'cup' generators, missing when the morphisms are linear maps, make it possible to model feedback. The relations say, among other things, that the 1-dimensional vector space k has two special commutative †-Frobenius structures, such that the multiplication and unit of either one and the comultiplication and counit of the other fit together to form a bimonoid. This sort of structure, but with tensor product replacing direct sum, is familiar from the 'ZX-calculus' obeyed by a finite-dimensional Hilbert space with two mutually un-

biased bases. In order to address controllability and observability, we construct the PROP $\mathtt{Stateful}_k$ and relate it back to the PROP of signal-flow diagrams. This provides a way to graphically express controllability and observability for linear time-invariant processes.

Contents

List of Figures

Chapter 1

Introduction

1.1 Outline

Control theory is the branch of engineering that focuses on manipulating 'open systems'—systems with inputs and outputs—to achieve desired goals. In control theory, several graphical models—*e.g.* 'signal-flow graphs' and 'box diagrams'—have been used to describe linear ways of manipulating signals, which we will take here to be smooth real-valued functions of time [14]. For a category theorist, at least, it is natural to treat these graphical models as string diagrams in a symmetric monoidal category [16, 17]. Here we use the term **signal-flow diagram** to refer to these string diagrams. This forces some small changes of perspective, which we discuss below, but more important is the question: *which symmetric monoidal category?*

We shall argue that a first approximation to the answer is: the category FinRel_k of finite-dimensional vector spaces over a certain field k, but with *linear relations* rather than linear maps as morphisms, and *direct sum* rather than tensor product providing the symmetric monoidal structure. We use the field $k = \mathbb{R}(s)$ consisting of rational functions in one real variable s. This variable has the meaning of differentation. A linear relation from k^m to k^n is thus a system of linear constant-coefficient ordinary differential equations relating m 'input' signals and n 'output' signals.

A second approximation to the answer is: the category $\mathrm{Stateful}_k$ of finite-dimensional vector spaces over a certain field $k(s)$ with 'stateful' morphisms which, roughly speaking, distinguish the paths that involve s from those that do not involve s. Now there are m 'inputs', n 'states', and p 'outputs'. When $k = \mathbb{R}$, we are again back to the situation of rational functions in one real variable s. This category is developed and discussed in Chapter 4. The key advantage to $\mathrm{Stateful}_k$ over $\mathrm{FinRel}_{k(s)}$ is the ability to extract the control theoretic concepts of controllability and observability from a stateful morphism. The key disadvantage is stateful morphisms evaluate to linear maps rather than linear relations. So while every signal-flow diagram has a linear relation associated to it, not every signal-flow diagram has a stateful morphism associated to it.

Our main goal for the first approximation is to provide a complete 'generators and equations' picture of this symmetric monoidal category, with the generators being familiar components of the graphical models used by control theorists. It turns out that the answer has an intriguing but mysterious connection to ideas that are familiar in the diagrammatic approach to quantum theory. Quantum theory also involves linear algebra, but it uses linear maps between Hilbert spaces as morphisms, and the tensor product of Hilbert spaces provides the symmetric monoidal structure.

For the second approximation, our main goal is to identify which signal-flow diagrams describe controllable (*resp.* observable) systems. It turns out that not all signal-flow diagrams admit as 'stateful' description, so part of this goal is the question, for which signal-flow diagrams can we ask about controllability and observability? We hope that the category-theoretic viewpoint on signal-flow diagrams will shed new light on control theory. However, in this dissertation we only lay the groundwork.

Briefly, the plan is as follows: Chapter 2 introduces the the machinery of PROPs, explaining how to describe a PROP using generators and equations and how to work with PROPs using this description. PROPs form a particularly simple class of symmetric monoidal categories that includes $\mathtt{FinRel}_k$ and $\mathtt{Stateful}_k$. By Mac Lane's coherence theorem [25], the PROPs $\mathtt{Stateful}_k$ and $\mathtt{FinRel}_k$ are equivalent to the categories $\mathrm{Stateful}_k$ and FinRel_k described above. This leads to Chapter 3, which gives a presentation of $\mathtt{FinRel}_k$, introduces signal-flow diagrams, and summarizes the main results of our first approximation. To get to the second approximation, Chapter 4 introduce a new PROP, $\mathtt{Stateful}_k$ and describes how it relates to $\mathtt{FinRel}_{k(s)}$. Chapter 4 also describes how to determine controllability and observability from a stateful morphism. The main result of the second approximation appears in Chapter 5, where we consider signal-flow diagrams as mathematical entities in their own right and describe the subcategory of signal-flow diagrams that admit a stateful description. For a signal-flow diagram that admit such a description, the description provides a path to determining controllability and observability for the signal-flow diagram. Finally, Chapter 6 deals with future work: we describe ways in which the stateful description can be extended to larger subcategories of signal-flow diagrams and how the category $\mathtt{Circ}$ of open passive electric circuits with linear circuit elements can be viewed as a category of signal-flow diagrams. This second direction for future work would connect the present work with that of Baez and Fong [4]. In the following sections we sketch some of the main ideas of this plan.

1.2 PROPs, linear relations, and signal-flow diagrams

In his famous thesis, Lawvere [24] introduced 'functorial semantics'. In this idea, a functor $F\colon \mathcal{C} \to \mathcal{D}$ sends formal expressions, which are morphisms in $\mathcal{C}$, to their 'meanings', which are morphisms in $\mathcal{D}$. One says that $\mathcal{C}$ provides the 'syntax' and $\mathcal{D}$ the 'semantics'. Here we apply this idea to control theory. For example, we may take $\mathcal{C}$ to be a category where morphisms are signal-flow diagrams, and $\mathcal{D}$ to be $\texttt{FinRel}_k$: then we shall construct a 'black-boxing functor' sending any signal-flow diagram to the linear relation it stands for.

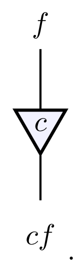
To apply Lawvere's ideas one wants categories equipped with extra structure: in our work we use **PROPs**, which are strict symmetric monoidal categories whose objects are natural numbers, the tensor product of objects being given by addition. In Chapter 2 we explain how to describe PROPs using generators and equations. This follows the work of Baez, Coya and Rebro [2], which is based on the work of Trimble [33]. Chapter 2 also has parallels in Zanasi's Ph.D. dissertation [37, Chap. 2.2].

Of key importance in the present work is the existence and uniqueness of a functor from the free PROP on some generators to a PROP presented by those generators and some equations. We continue in Chapter 3 with a generators and equations description of $\texttt{FinRel}_k$. This chapter can also be found in [3], with some minor changes made in the present work to streamline its connections to the other chapters. This begins the formalization into signal-flow diagrams of what control theorists do with their graphical models. When $k = \mathbb{R}(s)$, a morphisms in $\texttt{FinRel}_k$ describes the relation between some input signals and output signals, corresponding to what control theorists call 'transfer functions'.[1] The generators of $\texttt{FinRel}_k$ correspond to some of the most basic operations one might want to perform when manipulating signals. The simplest operation is amplification, or 'scaling': multiplying a signal by a scalar. A signal can be scaled by a constant factor:

$$f \mapsto cf,$$

where $c \in \mathbb{R}$. We can write this as a signal-flow diagram:

[1]Control theorists generally only deal with linear *maps* rather than linear *relations* in this context, so a pedant may argue for the invention of a new jargon term here, 'transfer relation'.

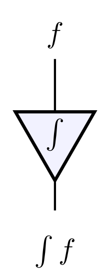


Here the labels f and cf on top and bottom are just for explanatory purposes and not really part of the diagram. Control theorists often draw arrows on the wires, but this is unnecessary from the string diagram perspective. Arrows on wires are useful to distinguish objects from their duals, but ultimately we will obtain a compact closed category where each object is its own dual, so the arrows can be dropped. What we really need is for the box denoting scalar multiplication to have a clearly defined input and output. This is why we draw it as a triangle. Control theorists often use a rectangle or circle, using arrows on wires to indicate which carries the input f and which the output cf.

A signal can also be integrated with respect to the time variable:

$$f \mapsto \int f.$$

Mathematicians typically take differentiation as fundamental, but engineers sometimes prefer integration, because it is more robust against small perturbations. In the end it will not matter much here. We can again draw integration as a signal-flow diagram:



Since this looks like the diagram for scaling, it is natural to extend $\mathbb{R}$ to $\mathbb{R}(s)$, the field of rational functions of a variable s which stands for differentiation. Then differentiation becomes a special case of scalar multiplication, namely multiplication by s, and integration becomes multiplication by $1/s$. Engineers accomplish the same effect with Laplace transforms, since differentiating a signal f is equivalent to multiplying its Laplace transform

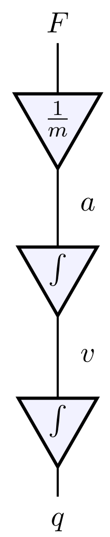
$$(\mathcal{L}f)(s) = \int_0^\infty f(t)e^{-st}\, dt$$

by the variable s. Another option is to use the Fourier transform: differentiating f is equivalent to multiplying its Fourier transform

$$(\mathcal{F}f)(\omega) = \int_{-\infty}^\infty f(t)e^{-i\omega t}\, dt$$

by $-i\omega$. Of course, the function f needs to be sufficiently well-behaved to justify calculations involving its Laplace or Fourier transform. At a more basic level, it also requires some work to treat integration as the two-sided inverse of differentiation. Engineers do this by considering signals that vanish for $t < 0$, and choosing the antiderivative that vanishes under the same condition. Luckily all these issues can be side-stepped in a formal treatment of signal-flow diagrams: we can simply treat signals as living in an unspecified vector space over the field $\mathbb{R}(s)$. The field $\mathbb{C}(s)$ would work just as well, and control theory relies heavily on complex analysis. In most of this paper we work over an arbitrary field k.

The simplest possible signal processor is a rock, which takes the 'input' given by the force F on the rock and produces as 'output' the rock's position q. Thanks to Newton's second law $F = ma$, we can describe this using a signal-flow diagram:



Here composition of morphisms is drawn in the usual way, by attaching the output wire of one morphism to the input wire of the next.

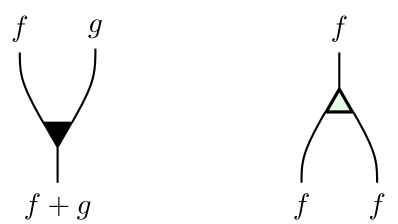
To build more interesting machines we need more building blocks, such as addition:

$$+\colon (f,g) \mapsto f+g$$

and duplication:

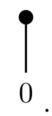
$$\Delta\colon f \mapsto (f,f).$$

When these linear maps are written as matrices, their matrices are transposes of each other. This is reflected in the signal-flow diagrams for addition and duplication:

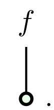


The second is essentially an upside-down version of the first. However, we draw addition as a dark triangle and duplication as a light one because we will later want another way to 'turn addition upside-down' that does *not* give duplication. As an added bonus, a light upside-down triangle resembles the Greek letter Δ, the usual symbol for duplication.

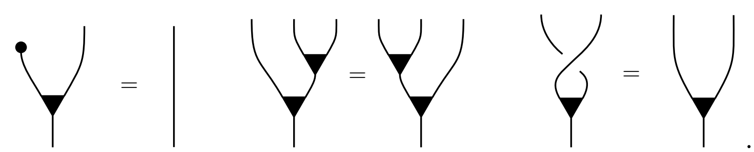
While they are typically not considered worthy of mention in control theory, for completeness we must include two other building blocks. One is the zero map from $\{0\}$ to our field k, which we denote as 0 and draw its signal-flow diagram as follows:



The other is the zero map from k to $\{0\}$, sometimes called 'deletion', which we denote as ! and draw thus:



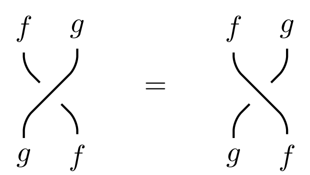
Just as the matrices for addition and duplication are transposes of each other, so are the matrices for zero and deletion, though they are rather degenerate, being 1×0 and 0×1 matrices, respectively. Addition and zero make k into a commutative monoid, meaning that the following equations hold:



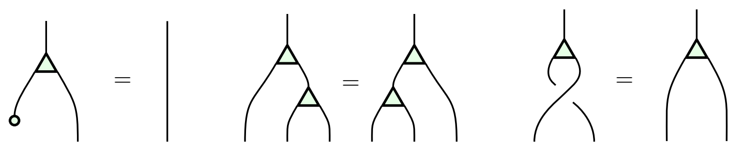
The equation at right is the commutative law, and the crossing of strands is the 'braiding'

$$B\colon (f, g) \mapsto (g, f)$$

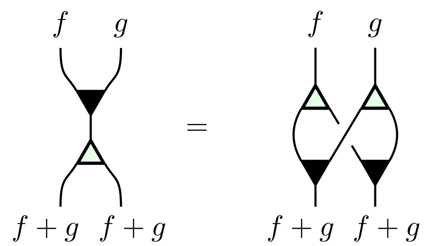
by which we switch two signals. In fact this braiding is a 'symmetry', so it does not matter which strand goes over which:



Dually, duplication and deletion make k into a cocommutative comonoid. This means that if we reflect the equations obeyed by addition and zero across the horizontal axis and turn dark operations into light ones, we obtain another set of valid equations:



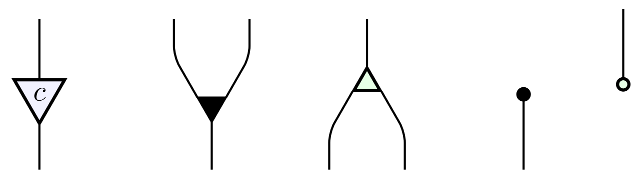
There are also equations between the monoid and comonoid operations. For example, adding two signals and then duplicating the result gives the same output as duplicating each signal and then adding the results:



This diagram is familiar in the theory of Hopf algebras, or more generally bialgebras. Here it is an example of the fact that the monoid operations on k are comonoid homomorphisms—or equivalently, the comonoid operations are monoid homomorphisms. We summarize this situation by saying that k is a **bimonoid**.

So far all our string diagrams denote linear maps. We can treat these as morphisms in the category $\mathrm{FinVect}_k$, where objects are finite-dimensional vector spaces over a field k and morphisms are linear maps. This category is equivalent to a skeleton where the only objects are vector spaces k^n for $n \geq 0$, and then morphisms can be seen as $n \times m$ matrices. This skeleton is actually a PROP. The space of signals is a vector space V over k which may not be finite-dimensional, but this does not cause a problem: an $n \times m$ matrix with entries in k still defines a linear map from V^n to V^m in a functorial way.

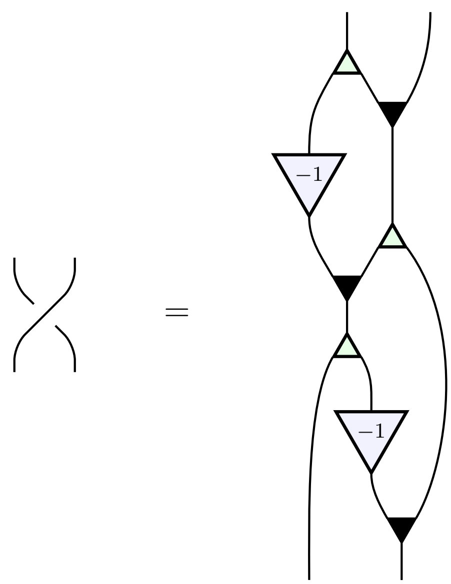
In applications of string diagrams to quantum theory [5, 11], we make $\mathrm{FinVect}_k$ into a symmetric monoidal category using the tensor product of vector spaces. In control theory, we instead make $\mathrm{FinVect}_k$ into a symmetric monoidal category using the *direct sum* of vector spaces. In Lemma 12 we prove that for any field k, $\mathrm{FinVect}_k$ with direct sum is generated as a symmetric monoidal category by the one object k together with these morphisms:



,

where $c \in k$ is arbitrary.

However, these generating morphisms obey some unexpected equations! For example, we have:



.

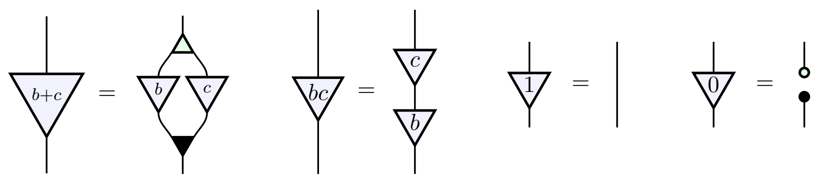
Thus, it is important to find a complete set of equations obeyed by these generating morphisms, thus obtaining a presentation of $\mathtt{FinVect}_k$ as a PROP. We do this in Theorem 13. In brief, these equations say:

1. $(k, +, 0, \Delta, !)$ is a bicommutative bimonoid;

2. the rig operations of k can be recovered from the generating morphisms;

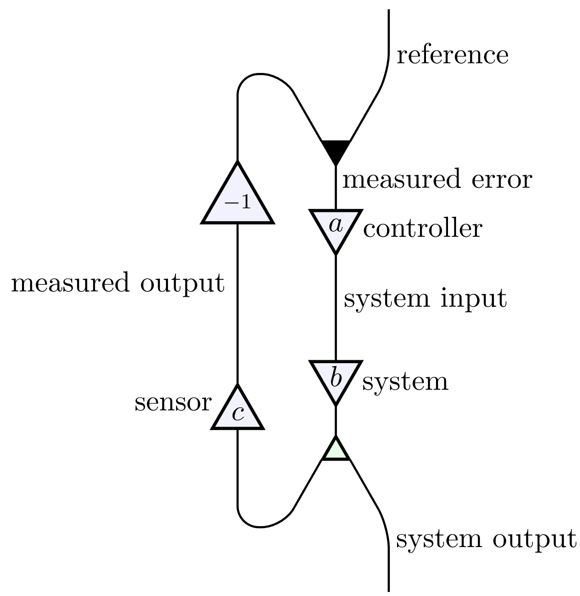
3. all the generating morphisms commute with scaling.

Here item (2) means that $+$, $\cdot$, 0 and 1 in the field k can be expressed in terms of signal-flow diagrams as follows:



Multiplicative inverses cannot be so expressed, so our signal-flow diagrams so far do not know that k is a field. Additive inverses also cannot be expressed in this way. And indeed, a version of Theorem 13 holds whenever k is a commutative rig: that is, a commutative 'ring without negatives', such as $\mathbb{N}$. The case of a commutative rig k was examined by Wadsley and Woods [35]: see Section 3.4 for details. The idea of finding a presentation for the category $\mathrm{FinVect}_k$ is not new. Indeed, Lafont [23] gave a presentation of $\mathrm{FinVect}_k$ as a monoidal category, with especial interest in the field of two elements, using generators and equations similar to the ones given here.

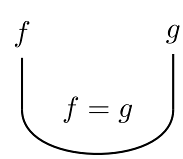
While Theorem 13 is a step towards understanding the category-theoretic underpinnings of control theory, it does not treat signal-flow diagrams that include 'feedback'. Feedback is one of the most fundamental concepts in control theory because a control system without feedback may be highly sensitive to disturbances or unmodeled behavior. Feedback allows these disturbances to be mollified (or exacerbated!). As an annotated string diagram, a basic feedback system might look like this:



.

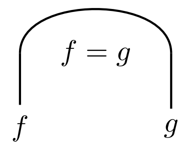
The user inputs a 'reference' signal, which is fed into a controller, whose output is fed into a system, or 'plant', which in turn produces its own output. But then the system's output is duplicated, and one copy is fed into a sensor, whose output is added[2] to the reference signal.

In string diagrams—unlike in the usual thinking on control theory—it is essential to be able to read any diagram from top to bottom as a composite of tensor products of generating morphisms. Thus, to incorporate the idea of feedback, we need two more generating morphisms. These are the 'cup':



and 'cap':

[2]More typically this output is subtracted in controlled systems, since disturbances are frequently unwanted.



.

These are not maps; they are relations. The cup imposes the relation that its two inputs be equal, while the cap does the same for its two outputs. This is a way of describing how a signal flows around a bend in a wire.

To make this precise, we use a category called FinRel_k. An object of this category is a finite-dimensional vector space over k, while a morphism from U to V, denoted $L\colon U \nrightarrow V$, is a **linear relation**, meaning a linear subspace

$$L \subseteq U \oplus V.$$

In particular, when $k = \mathbb{R}(s)$, a linear relation $L\colon k^m \to k^n$ is just an arbitrary system of constant-coefficient linear ordinary differential equations relating m input variables and n output variables.

Since the direct sum $U \oplus V$ is also the cartesian product of U and V, a linear relation is indeed a relation in the usual sense, but with the property that if $u \in U$ is related to $v \in V$ and $u' \in U$ is related to $v' \in V$ then $cu + c'u'$ is related to $cv + c'v'$ whenever $c, c' \in k$. We compose linear relations $L\colon U \nrightarrow V$ and $L'\colon V \nrightarrow W$ as follows:

$$L'L = \{(u, w)\colon\ \exists\, v \in V\ \ (u, v) \in L \text{ and } (v, w) \in L'\}.$$

Any linear map $f\colon U \to V$ gives a linear relation $F\colon U \nrightarrow V$, namely the graph of that map:

$$F = \{(u, f(u)) : u \in U\}.$$

Composing linear maps thus becomes a special case of composing linear relations, so $\mathrm{FinVect}_k$ becomes a subcategory of FinRel_k. Furthermore, we can make FinRel_k into a monoidal category using direct sums, and it becomes symmetric monoidal using the braiding already present in $\mathrm{FinVect}_k$.

In these terms, the **cup** is the linear relation

$$\cup\colon k^2 \nrightarrow \{0\}$$

given by

$$\cup \;=\; \{(x, x, 0) : x \in k\} \;\subseteq\; k^2 \oplus \{0\},$$

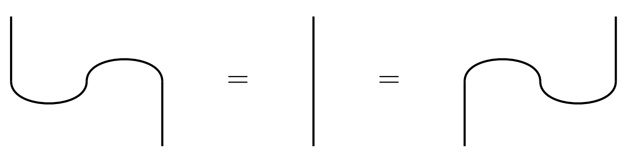
while the **cap** is the linear relation

$$\cap\colon \{0\} \nrightarrow k^2$$

given by

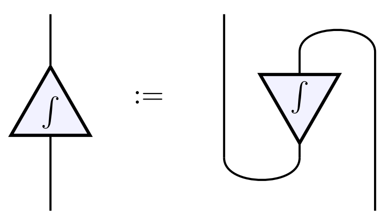
$$\cap \;=\; \{(0, x, x) : x \in k\} \;\subseteq\; \{0\} \oplus k^2.$$

These obey the **zigzag equations**:



Thus, they make FinRel_k into a compact closed category where k, and thus every object, is its own dual. As with $\mathrm{FinVect}_k$, we will focus on a skeleton $\mathtt{FinRel}_k$ of FinRel_k, which is a PROP.

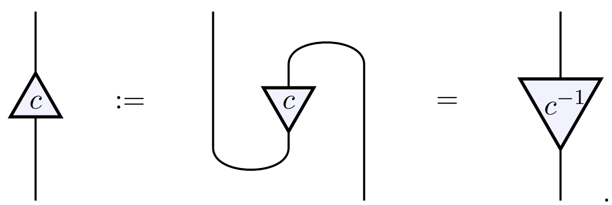
Besides feedback, one of the things that make the cap and cup useful is that they allow any morphism $L\colon U \nrightarrow V$ to be 'plugged in backwards' and thus 'turned around'. For instance, turning around integration:



we (essentially) obtain differentiation. In general, using caps and cups we can turn around any linear relation $L\colon U \nrightarrow V$ and obtain a linear relation $L^\dagger\colon V \nrightarrow U$, called the **adjoint** of L, which turns out to given by

$$L^\dagger = \{(v,u) : (u,v) \in L\}.$$

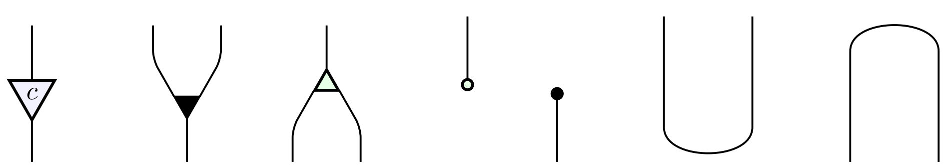
For example, if $c \in k$ is nonzero, the adjoint of scalar multiplication by c is multiplication by c^{-1}:



Thus, caps and cups allow us to express multiplicative inverses in terms of signal-flow diagrams! One might think that a problem arises when when $c = 0$, but no: the adjoint of scaling by 0 is the linear relation

$$\{(0,x) : x \in k\} \subseteq k \oplus k.$$

In Lemma 14 we show that $\mathtt{FinRel}_k$ is generated, as a symmetric monoidal category, by these morphisms:



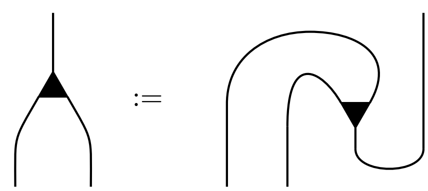
where $c \in k$ is arbitrary.

In Theorem 15 we find a complete set of equations obeyed by these generating morphisms, thus giving a presentation of $\mathtt{FinRel}_k$ as a PROP. To describe these equations, it is useful to work with adjoints of the generating morphisms. We have already seen that the adjoint of scaling by c is scaling by c^{-1}, except when $c = 0$. Taking adjoints of the other four generating morphisms of $\mathtt{FinVect}_k$, we obtain four important but perhaps unfamiliar linear relations. We draw these as 'turned around' versions of the original generating morphisms:

- **Coaddition** is a linear relation from k to k^2 that holds when the two outputs sum to the input:

$$+^\dagger \colon k \nrightarrow k^2$$

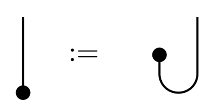
$$+^\dagger = \{(x, y, z) : \ x = y + z\} \subseteq k \oplus k^2$$



- **Cozero** is a linear relation from k to $\{0\}$ that holds when the input is zero:

$$0^\dagger \colon k \nrightarrow \{0\}$$

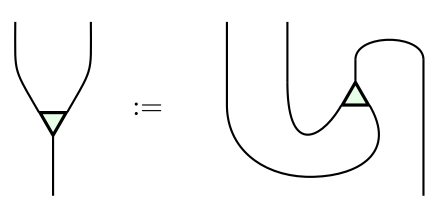
$$0^\dagger = \{(0, 0)\} \subseteq k \oplus \{0\}$$



- **Coduplication** is a linear relation from k^2 to k that holds when the two inputs both equal the output:

$$\Delta^\dagger \colon k^2 \nrightarrow k$$

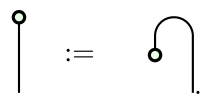
$$\Delta^\dagger = \{(x, y, z) : \ x = y = z\} \subseteq k^2 \oplus k$$



- **Codeletion** is a linear relation from $\{0\}$ to k that holds always:

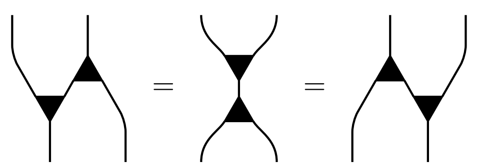
$$!^\dagger \colon \{0\} \nrightarrow k$$

$$!^\dagger = \{(0, x)\} \subseteq \{0\} \oplus k$$



Since $+^\dagger, 0^\dagger, \Delta^\dagger$ and $!^\dagger$ automatically obey turned-around versions of the equations obeyed by $+, 0, \Delta$ and $!$, we see that k acquires a *second* bicommutative bimonoid structure when considered as an object in FinRel_k.

Moreover, the four dark operations make k into a **Frobenius monoid**. This means that $(k, +, 0)$ is a monoid, $(k, +^\dagger, 0^\dagger)$ is a comonoid, and the **Frobenius relation** holds:

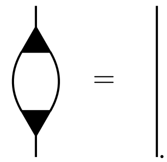


All three expressions in this equation are linear relations saying that the sum of the two inputs equal the sum of the two outputs.

The operation sending each linear relation to its adjoint extends to a contravariant functor

$$\dagger \colon \mathrm{FinRel}_k \to \mathrm{FinRel}_k,$$

which obeys a list of properties that are summarized by saying that FinRel_k is a '$\dagger$-compact' category [1, 28]. Because two of the operations in the Frobenius monoid $(k, +, 0, +^\dagger, 0^\dagger)$ are adjoints of the other two, it is a **†-Frobenius monoid**. This Frobenius monoid is also **special**, meaning that comultiplication (in this case $+^\dagger$) followed by multiplication (in this case $+$) equals the identity on k:



This Frobenius monoid is also commutative—and cocommutative, but for Frobenius monoids this follows from commutativity.

Starting around 2008, commutative special †-Frobenius monoids have become important in the categorical foundations of quantum theory, where they can be understood as 'classical structures' for quantum systems [12, 34]. The category FinHilb of finite-dimensional Hilbert spaces and linear maps is a †-compact category, where any linear map $f\colon H \to K$ has an adjoint $f^\dagger\colon K \to H$ given by

$$\langle f^\dagger \phi, \psi \rangle = \langle \phi, f\psi \rangle$$

for all $\psi \in H, \phi \in K$. A commutative special †-Frobenius monoid in FinHilb is then the same as a Hilbert space with a chosen orthonormal basis. The reason is that given an orthonormal basis ψ_i for a finite-dimensional Hilbert space H, we can make H into a commutative special †-Frobenius monoid with multiplication $m\colon H \otimes H \to H$ given by

$$m(\psi_i \otimes \psi_j) = \begin{cases} \psi_i & i = j \\ 0 & i \neq j \end{cases}$$

and unit $i\colon \mathbb{C} \to H$ given by

$$i(1) = \sum_i \psi_i.$$

The comultiplication $m^\dagger$ duplicates basis states:

$$m^\dagger(\psi_i) = \psi_i \otimes \psi_i.$$

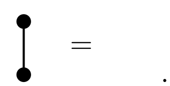
Conversely, any commutative special †-Frobenius monoid in FinHilb arises this way.

Considerably earlier, around 1995, commutative Frobenius monoids were recognized as important in topological quantum field theory. The reason, ultimately, is that the free symmetric monoidal category on a commutative Frobenius monoid is 2Cob, the category with 2-dimensional oriented cobordisms as morphisms: see Kock's textbook [20] and the many references therein. But the free symmetric monoidal category on a commutative *special* Frobenius monoid was worked out even earlier [9, 21, 27]: it is the category with finite sets as objects, where a morphism $f\colon X \to Y$ is an isomorphism class of cospans

$$X \longrightarrow S \longleftarrow Y.$$

This category can be made into a †-compact category in an obvious way, and then the 1-element set becomes a commutative special †-Frobenius monoid.

For all these reasons, it is interesting to find a commutative special †-Frobenius monoid lurking at the heart of control theory! However, the Frobenius monoid here has yet another property, which is more unusual. Namely, the unit $0\colon \{0\} \nrightarrow k$ followed by the counit $0^\dagger\colon k \nrightarrow \{0\}$ is the identity on $\{0\}$:



We call a special Frobenius monoid that also obeys this 'extra' law **extra-special**. One can check that the free symmetric monoidal category on a commutative extra-special Frobenius monoid is the category with finite sets as objects, where a morphism $f\colon X \to Y$ is an equivalence relation on the disjoint union $X \sqcup Y$, and we compose $f\colon X \to Y$ and $g\colon Y \to Z$ by letting f and g generate an equivalence relation on $X \sqcup Y \sqcup Z$ and then restricting this to $X \sqcup Z$.

As if this were not enough, the light operations share many properties with the dark ones. In particular, these operations make k into a commutative extra-special †-Frobenius monoid in a second way. In summary:

- $(k, +, 0, \Delta, !)$ is a bicommutative bimonoid;
- $(k, \Delta^\dagger, !^\dagger, +^\dagger, 0^\dagger)$ is a bicommutative bimonoid;

- $(k, +, 0, +^\dagger, 0^\dagger)$ is a commutative extra-special †-Frobenius monoid;

- $(k, \Delta^\dagger, !^\dagger, \Delta, !)$ is a commutative extra-special †-Frobenius monoid.

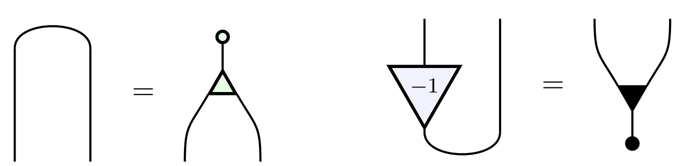
It should be no surprise that with all these structures built in, signal-flow diagrams are a powerful method of designing processes. However, it is surprising that most of these structures are present in a seemingly very different context: the so-called 'ZX calculus', a diagrammatic formalism for working with complementary observables in quantum theory [10]. This arises naturally when one has an n-dimensional Hilbert space H with two orthonormal bases ψ_i, ϕ_i that are 'mutually unbiased', meaning that

$$|\langle \psi_i, \phi_j \rangle|^2 = \frac{1}{n}$$

for all $1 \leq i, j \leq n$. Each orthonormal basis makes H into commutative special †-Frobenius monoid in FinHilb. Moreover, the multiplication and unit of either one of these Frobenius monoids fits together with the comultiplication and counit of the other to form a bicommutative bimonoid. So, we have all the structure present in the list above—except that these Frobenius monoids are only *extra*-special if H is 1-dimensional.

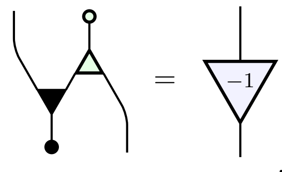
The field k is also a 1-dimensional vector space, but this is a red herring: in $\texttt{FinRel}_k$ *every* finite-dimensional vector space naturally acquires all four structures listed above, since addition, zero, duplication and deletion are well-defined and obey all the equations we have discussed. We focus on k in this paper simply because it generates all the objects $\texttt{FinRel}_k$ via direct sum.

Finally, in $\texttt{FinRel}_k$ the cap and cup are related to the light and dark operations as follows:



.

Note the curious factor of -1 in the second equation, which breaks some of the symmetry we have seen so far. This equation says that two elements $x, y \in k$ sum to zero if and only if $-x = y$. Using the zigzag equations, the two equations above give the antipode



We thus see that in $\texttt{FinRel}_k$, both additive and multiplicative inverses can be expressed in terms of the generating morphisms used in signal-flow diagrams.

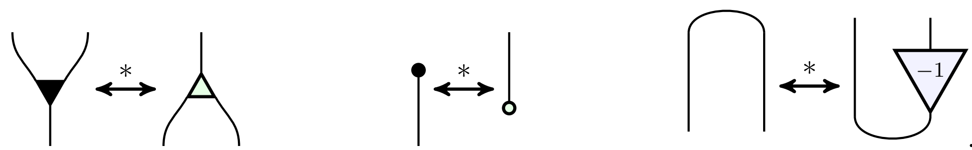
The break in symmetry at this point can be explained by yet another second way of doing something. We have seen one contravariant functor on $\texttt{FinRel}_k$, $\dagger$, but there is a second contravariant functor on $\texttt{FinRel}_k$, $*$. This one extends a contravariant functor on $\texttt{FinVect}_k$ that was already lurking in the background. The functor

$$*\colon \texttt{FinRel}_k \to \texttt{FinRel}_k$$

extends the notion of *transposition* of linear maps, and these two equations relating the cap and cup to light and dark operations show how to consistently extend transposition to cap and cup, and thus to linear relations. Thus we have

- $+^* = \Delta$,
- $\Delta^* = +$,
- $0^* = !$,
- $!^* = 0$,
- $\cap^* = \cup \circ (1 \oplus s_{-1})$,
- $\cup^* = (1 \oplus s_{-1}) \circ \cap$.

Graphically,



Theorem 15 gives a presentation of $\mathtt{FinRel}_k$ based on some of the ideas just discussed. Briefly, it says that $\mathtt{FinRel}_k$ is the PROP generated by these morphisms:

1. addition $+\colon k^2 \nrightarrow k$
2. zero $0\colon \{0\} \nrightarrow k$
3. duplication $\Delta\colon k \nrightarrow k^2$
4. deletion $!\colon k \nrightarrow 0$
5. scaling $s_c\colon k \nrightarrow k$ for any $c \in k$
6. cup $\cup\colon k^2 \nrightarrow \{0\}$
7. cap $\cap\colon \{0\} \nrightarrow k^2$

obeying these equations:

1. $(k, +, 0, \Delta, !)$ is a bicommutative bimonoid;
2. $\cap$ and $\cup$ obey the zigzag equations;
3. $(k, +, 0, +^\dagger, 0^\dagger)$ is a commutative extra-special †-Frobenius monoid;
4. $(k, \Delta^\dagger, !^\dagger, \Delta, !)$ is a commutative extra-special †-Frobenius monoid;
5. the field operations of k can be recovered from the generating morphisms;
6. the generating morphisms (1)–(4) commute with scaling.

Note that item (2) makes $\texttt{FinRel}_k$ into a †-compact category, allowing us to mention the adjoints of generating morphisms in the subsequent equations. Item (5) means that $+, \cdot, 0, 1$ and also additive and multiplicative inverses in the field k can be expressed in terms of signal-flow diagrams in the manner we have explained.

1.3 State space

Control theory underwent a paradigm shift in the 1960s with the advent of the state-space approach. Chapter 4 introduces the basic ideas of this approach and builds up to the PROP $\texttt{Stateful}_k$, which we designed to describe this approach more closely than $\texttt{FinRel}_k$ can.

The state-space approach to control theory was born around 1960 with Kalman's paper [18] that introduced to the world the concepts of controllability and observability. This approach addresses some of the limitations of the frequency analysis approach, which had enjoyed significant early success. Kalman noticed any linear time-invariant (LTI) control system can be partitioned into four subsystems[3], only one of which is accounted for by the transfer function of the frequency analysis approach. The other three subsystems lack inputs, lack outputs, or lack both, thus are best studied by looking at the internal states of a system. The continuous time version of the state-space approach uses matrix differential equations that involve the input and output of a system, mediated by the internal state of the system. In a linear time-invariant system, which is the only kind we consider, these equations are

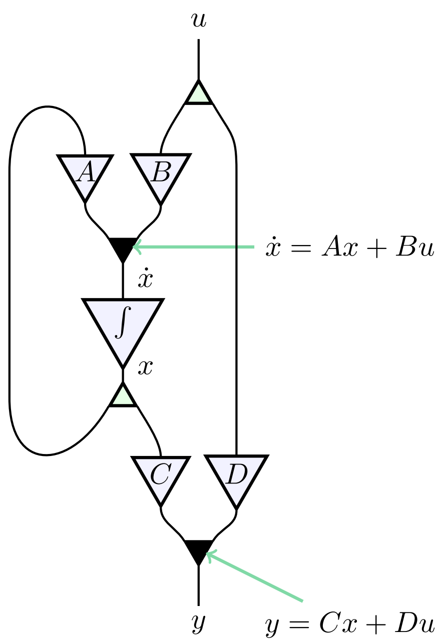
$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1.1}$$

$$y(t) = Cx(t) + Du(t), \tag{1.2}$$

[3]This partitioning can also be done for nonlinear or time-varying systems, but the four parts are no longer necessarily control systems in their own right.

where $u(t)$ is the input vector, $y(t)$ is the output vector, and $x(t)$ is the state vector. These equations can also be discretized to matrix difference equations for a discrete time approach. Unless otherwise stated, we will use the convention that $\dim(u) = m$, $\dim(x) = n$, and $\dim(y) = p$.

Equations 1.1 and 1.2 can be found lurking in the following signal-flow diagram:



where we have used the shorthand of drawing a single generating morphism where there are zero or more parallel generating morphisms of the same kind and scaling representing matrix multiplication. Note that taking integration to be scaling by $\frac{1}{s}$, as when taking Laplace transforms, the linear relation this signal-flow diagram depicts is the linear map $D + C(sI - A)^{-1}B$.

A system is **controllable** if for each state x and time t_0 there is an input function $u(t)$ such that the state can be set to the equilibrium state, *i.e.* the zero vector, in a finite amount of time. For the linear time-invariant systems we are interested in, there is a simple characterization of controllability involving the row rank of the block matrix $M_c = [B, AB, \ldots, A^{n-1}B]$. This controllability matrix M_c is an $n \times mn$ matrix, and a system is controllable when its row rank is n:

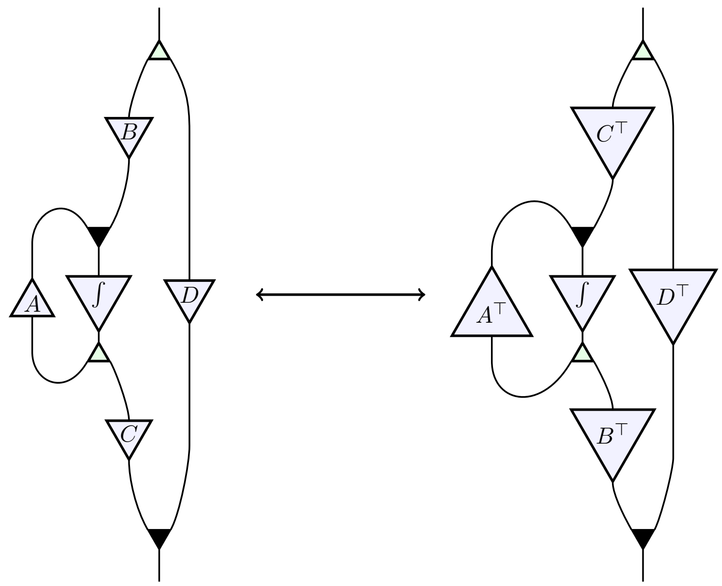
$$\operatorname{rank}(M_c) = n.$$

A system is **observable** if for each state x and time t_0, and with the input function $u(t)$ identically zero, measurements of the output function $y(t)$ over a finite duration can be used to determine the state $x(t_0)$. For the systems we are concerned with, there is a characterization of observability in terms of the column rank of the block matrix $M_o = [C, CA, \ldots, CA^{n-1}]^\top$. This observability matrix M_o is an $np \times n$ matrix, and a linear time-invariant system is observable when its column rank is n:

$$\operatorname{rank}(M_o) = n.$$

There are clear parallels in these descriptions of controllability and observability, but there is a seeming fly in the ointment with observability depending on the input signal being zero and controllability being independent of the output signal. Despite this oddity, it is not difficult to guess there might be some kind of duality relating controllability and observability. Indeed, Kalman defined observability in [18] as a dual notion to controllability, and only defined it as a separate concept later. The action of Kalman's duality reverses the direction of time, swaps the roles of the matrices B and C, and transposes all the matrices A, B, C, and D. Even in the time-varying case, this process transforms a controllable system into an observable system, and an observable system into a controllable system.

It is curious to see what happens when Kalman's duality is applied to the signal-flow diagram above that encodes the state-space equations.



Recalling that the transposition duality $*\colon \texttt{FinRel}_k \to \texttt{FinRel}_k$ vertically flips signal-flow diagrams and reverses the colors of the generators, Kalman's duality bears remarkable resemblance to the transposition duality. The similarity to the transposition duality can even be used to explain the oddity of controllability ignoring (deleting) the output signal and observability setting the input signal to zero: $!^* = 0$.

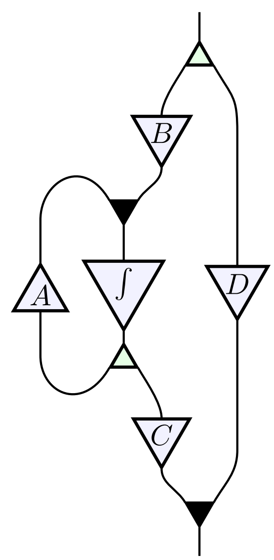
While it is clear something connects Kalman's work on controllability and observability to the PROP $\texttt{FinRel}_k$, taking the signal-flow diagrams above to be linear relations hides the evidence of the connection: it is impossible to reconstruct A, B, C, and D from a given linear relation. To deal with this shortcoming, we form a new PROP, $\square(\texttt{FinVect}_k)$, as a stepping stone towards finding the PROP $\texttt{Stateful}_k$. The objects of $\square(\texttt{FinVect}_k)$ are the vector spaces k^n just as with $\texttt{FinVect}_k$, but the morphisms from V_1 to V_2 are now 4-tuples of linear maps, which can be conveniently organized as non-commutative squares:

$$\begin{tikzcd} S \arrow[r, "a"] & T \arrow[d, "c"] \\ V_1 \arrow[u, "b"] \arrow[r, "d"'] & V_2 \end{tikzcd}.$$

For compactness of notation, this square can also be written (d, c, a, b).

In Theorem 18 we show there is an evaluation functor $\mathrm{eval}\colon \square(\mathtt{FinVect}_k) \to \mathtt{FinVect}_k$ that takes (d, c, a, b) to $d + cab$. Even better, eval is a PROP morphism. As noted above, the signal-flow diagram that encodes the state-space equations (Equations 1.1 and 1.2) gives a linear map, $D+C(sI-A)^{-1}B$. The maps D, C, A, and B are all morphisms in $\mathtt{FinVect}_k$ in the linear time-invariant case, so this looks very similar to the evaluation of a $\square(\mathtt{FinVect}_k)$ morphism.

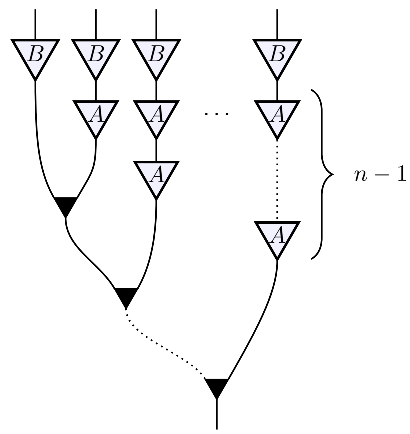
To get them to match, we define $\mathtt{Stateful}_k$ as a subPROP of $\square(\mathtt{FinVect}_{k(s)})$, where $d = D$, $c = C$, $a = (sI - A)^{-1}$, and $b = B$ for some linear maps A, B, C, and D. In Proposition 20 we show $\mathtt{Stateful}_k$ is a PROP. Given a stateful morphism (d, c, a, b), it is possible to find the linear maps A, B, C, and D used in the state-space equations. Because $\mathrm{eval}(d, c, a, b) = D + C(sI - A)^{-1}B$ for stateful morphisms, it is reasonable to allow the signal-flow diagram



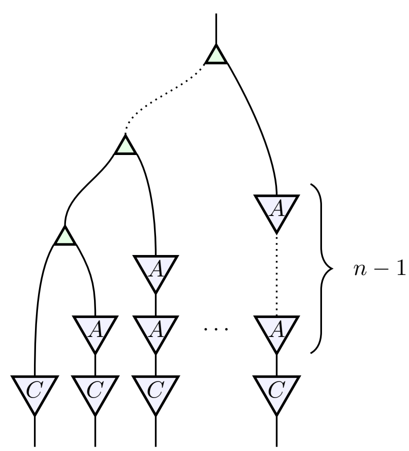
to depict a stateful morphism, not just a linear relation. Furthermore, because it is possible to find the linear maps A, B, C, and D used in the state-space equations, controllability and observability are well-defined for stateful morphisms. This gives a sense in which $\mathtt{Stateful}_k$ is a more detailed picture of a signal processing apparatus which captures not only the linear relation between inputs and outputs, but how the apparatus implements this relation.

In category theoretic terms, a linear map having full row rank means it is an epimorphism, and having full column rank means it is a monomorphism. We can therefore translate the linear time-invariant conditions for controllability and observability into signal-flow diagram form as follows:

A stateful morphism $(D, C, (sI - A)^{-1}, B)$ is controllable when



is an epimorphism in $\mathtt{FinVect}_k$, and it is observable when

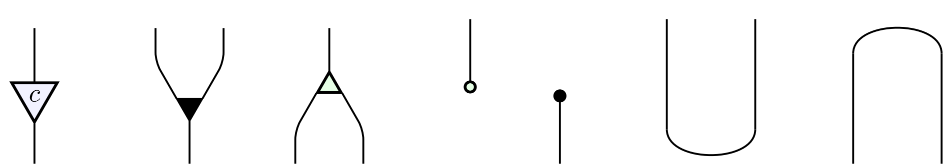


is a monomorphism in $\mathtt{FinVect}_k$.

Much of what has been discussed to this point has parallels in other contemporary work. Bonchi, Sobociński and Zanasi [6, 7] built up a similar generators and equations picture of $\mathbb{SV}_k$, a PROP which is identical to our $\texttt{FinRel}_k$, using Lack's idea [22] of composing PROPs. Sobociński also continued by considering controllability, but again from a different perspective: About 30 years after Kalman gave his definitions of controllability and observability, Willems [36] proposed alternative definitions for controllability and observability that are based on the behavior of a system. However, the duality between controllability and observability is less apparent in Willems' definition than in Kalman's definition. Nevertheless, Willems' behavioral approach is very fruitful, and Fong, Rapisarda and Sobociński [13] use this alternative definition to give a categorical characterization of behavioral controllability.

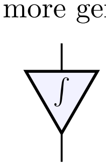
1.4 Controllability and observability in signal-flow diagrams

Signal-flow diagrams can do much more than depict linear relations. In Chapter 5 our goal is to define a PROP where the morphisms are the signal-flow diagrams used by control theorists, for which the all-important notions of controllability and observability, which we saw in the previous section, can be defined. We begin by defining a preliminary free PROP $\texttt{SigFlow}_k$, where morphisms are all diagrams that can be built up by these generators:



,

where $c \in k$ is arbitrary. Appending one more generator for integration



extends $\texttt{SigFlow}_k$ to a larger free PROP, $\texttt{SigFlow}_{k,s}$. All together, these are the generators of $\texttt{FinRel}_{k(s)}$ with the element s treated separately, since integrators play a special role in in control theory. $\mathtt{SigFlow}_{k,s}$ is simply the free prop on these generators. There is thus a morphism of props

$$\blacksquare\colon \mathtt{SigFlow}_{k,s} \to \mathtt{FinRel}_{k(s)}$$

sending each signal flow diagram to the linear relation between inputs and outputs that it determines. We call this the 'black-boxing' functor.

However, many morphisms in $\mathtt{SigFlow}_{k,s}$ are not signal-flow diagrams of the sort used in control theory; for example, one never sees the 'cup' or 'cap' above all by itself in a textbook on control theory. The challenge, then, is to pick out a subPROP $\mathtt{ContFlow}_k$ which consist of 'reasonable' signal-flow diagrams, for which controllability and observability can be defined.

We already have a category $\mathtt{Stateful}_k$ for which controllability and observability of morphisms can be defined, and in Section 4.3 we constructed a functor $\mathrm{eval}\colon \mathtt{Stateful}_k \to \mathtt{FinVect}_{k(s)}$. Composing with the inclusion $i\colon \mathtt{FinVect}_{k(s)} \to \mathtt{FinRel}_{k(s)}$ gives us a PROP morphism

$$i \circ \mathrm{eval}\colon \mathtt{Stateful}_k \to \mathtt{FinRel}_{k(s)}.$$

We would thus like $\mathtt{ContFlow}_k$ to be a PROP equipped with an inclusion $j\colon \mathtt{ContFlow}_k \to \mathtt{SigFlow}_{k,s}$ making this square commute:

$$\begin{tikzcd}
\mathtt{ContFlow}_k \arrow[r, "\Diamond"] \arrow[d, hook, "j"'] & \mathtt{Stateful}_k \arrow[d, "i \circ \mathrm{eval}"] \\
\mathtt{SigFlow}_{k,s} \arrow[r, "\blacksquare"'] & \mathtt{FinRel}_{k(s)}
\end{tikzcd}.$$

In fact, this desire will lead us directly to the definition of the PROP $\mathtt{ContFlow}_k$ in Definition 31. We conclude by showing some of the duality properties of $\mathtt{ContFlow}_k$ and how they are related to Kalman's duality between controllability and observability.

1.5 The 'Box' construction

In Appendix A we offer diagrammatic proofs of some derived equations used in the proof of Theorem 15. Some other diagrammatic proofs with miscellaneous connections are also included to indicate a portion of the richness of the connection between Frobenius bimonoids and bicommutative bimonoids. In Appendix B we expand on the 'Box' construction that led us to $\mathtt{Stateful}_k$ in Chapter 4.

When we first examine the Box construction in Chapter 4, we only apply it to the PROP $\mathtt{FinVect}_k$. The idea behind the Box construction of breaking up a morphism into the direct and indirect influences of the input on the output generalizes to a broader class of categories. It is straightforward to extend the Box construction to apply to the category $\mathrm{FinVect}_k$, or any category that has biproducts. What is exciting for the purposes of future work is that the Box construction can also be extended to apply to $\mathtt{FinRel}_k$ and FinRel_k. The key property of FinRel_k that makes it work is that $\mathrm{FinVect}_k$ is an **essentially wide** subcategory of FinRel_k. That is, $\mathrm{FinVect}_k$ 'essentially' contains all the objects of FinRel_k. More precisely, the inclusion functor $i\colon \mathrm{FinVect}_k \to \mathrm{FinRel}_k$ is essentially surjective.

Since $\mathrm{FinVect}_k$ has biproducts, every object in $\mathrm{FinVect}_k$ is a bicommutative bimonoid and every morphism is a bimonoid homomorphism. Thus every object in FinRel_k is a bicommutative bimonoid as well. In the Box construction in Chapter 4 we took advantage of the other fact, that all morphisms of $\mathtt{FinVect}_k$ are bimonoid homomorphisms. This is no longer the case in FinRel_k, but not all the arrows in the Box of a category need to be bimonoid homomorphisms. This opens up the possibility for a more general version of $\mathtt{Stateful}_k$, where a stateful morphism (d, c, a, b) is made up of linear relations a, b, c, and d, instead of simply linear maps. Using the same string diagram criteria for controllability and observability on the more general version of $\mathtt{Stateful}_k$ could potentially generalize the notions of controllability and observability in a way that has not been capitalized on in control theory.

Chapter 2

Generators and equations for PROPs

The formalism developed in this chapter gives us a way to present PROPs in an analogous way to the presentation of groups, where elements in a group are the analog to morphisms in a PROP. The signal-flow diagrams of control theory that appear throughout this dissertation fit into the convenient framework formed by PROPs for formalizing such diagrammatic techniques. Whereas a group is presented by a set of generators and a set of relations, a PROP is presented by a *distinguished object*, together with a *signature* which can be thought of as a collection of morphisms that generate the homsets, and a set of *equations* between elements of the same homset. Stated slightly differently, a PROP is presented by a distinguished object, a collection of generating morphisms, and a collection of equations. Before we do anything with PROPs, it would be good to say what a PROP is.

Definition 1 *A* **PROP** *is a strict symmetric monoidal category for which objects are natural numbers and the monoidal product is addition. A* **PROP morphism** *is a strict symmetric monoidal functor that maps the object 1 to the object 1.*

Stated this way, the distinguished object is the natural number 1. We note by Mac Lane's coherence theorem [25] that any symmetric monoidal category is equivalent to a strict symmetric monoidal category. We make the convention of using roman typeface (as in FinRel_k) for names of symmetric monoidal categories that may not be strict and typewriter typeface (as in $\mathtt{FinRel}_k$) for names of strict symmetric monoidal categories. Other typefaces are used for categories which, for the purposes of discussion, need not be symmetric monoidal categories. In subsequent chapters it will be convenient to think of the objects of a PROP as tensor powers of a distinguished object, X, using the one-to-one correspondence $X^{\otimes n} \mapsto n \in \mathbb{N}$. For example, we will be concerned with PROPs that have the vector spaces k^n over some field k as their objects, direct sum as tensor, and the one-dimensional vector space k as the distinguished object. There is a category PROP of PROPs and PROP morphisms.

Definition 2 *A* **symmetric monoidal theory** $T = (\Sigma, E)$ *is a signature* Σ *together with a set* E *of equations. A* **signature** *is a set of formal symbols* $\sigma\colon m \to n$*, where* $m, n \in \mathbb{N}$*. From a signature* Σ *we may formally construct the set of* Σ*-terms. Defined inductively, a* **Σ-term** *takes one of the following forms:*

- *the unit* $\mathrm{id}\colon 1 \to 1$*, the braiding* $\mathrm{b}\colon 2 \to 2$*, or the formal symbols* $\sigma\colon m \to n$ *in* Σ*;*
- $\beta \circ \alpha\colon m \to p$*, where* $\alpha\colon m \to n$ *and* $\beta\colon n \to p$ *are* Σ*-terms; or*
- $\alpha + \gamma\colon m + p \to n + q$*, where* $\alpha\colon m \to n$ *and* $\gamma\colon p \to q$ *are* Σ*-terms.*

We call (m, n) *the* **type** *of a* Σ*-term* $\alpha\colon m \to n$*. An* **equation** *is an ordered pair of* Σ*-terms with the same type.*

We can think of the type as an object in the discrete category $\mathbb{N} \times \mathbb{N}$. Then a signature is a functor from $\mathbb{N} \times \mathbb{N}$ to Set; to each type $(m, n) \in \mathbb{N} \times \mathbb{N}$, a signature assigns a set of formal symbols of that type. Note that each PROP $\mathtt{P}$ has an **underlying signature**, given by the functor $\hom_{\mathtt{P}}(\cdot, \cdot)\colon \mathbb{N} \times \mathbb{N} \to \mathrm{Set}$. The following result of Baez, Coya, and Rebro [2], building on the work of Trimble [33], allows us to understand the category PROP.

Proposition 3 *The underlying signature functor* $\mathtt{U}\colon \mathsf{PROP} \to \mathrm{Set}^{\mathbb{N}\times\mathbb{N}}$ *is monadic.*

By saying $\mathtt{U}$ is **monadic** we mean $\mathtt{U}$ has a right adjoint $\mathtt{F}\colon \mathrm{Set}^{\mathbb{N}\times\mathbb{N}} \to \mathsf{PROP}$, and the resulting functor from PROP to the category of algebras of the monad $\mathtt{FU}$ is an equivalence of categories [8]. We call $\mathtt{F}\Sigma$ the free PROP on the signature Σ. In fact, any Σ-term determines a morphism in $\mathtt{F}\Sigma$, and all morphisms in $\mathtt{F}\Sigma$ arise this way. For a Σ-term $\alpha\colon m \to n$, we abuse notation and refer to the corresponding morphism as $\alpha \in \hom(X^{\otimes m}, X^{\otimes n})$. For each formal symbol $\sigma\colon m \to n$ in Σ, we refer to its corresponding morphism σ as a **generator** for the free PROP on Σ.

Another important consequence of this proposition is that PROP is cocomplete. This guarantees the existence of coequalizers, which we use to construct a PROP for a symmetric monoidal theory.

Let (Σ, E) be a symmetric monoidal theory. Then E determines a signature $\mathcal{E}$, where each ordered pair in E determines a formal symbol in $\mathcal{E}$ whose type is the same as the type of the pair. We can define PROP morphisms $\lambda, \rho\colon \mathtt{F}\mathcal{E} \to \mathtt{F}\Sigma$ mapping the $\mathtt{F}$-image of each equation to the $\mathtt{F}$-images of the first element and second element of the pair, respectively.

Definition 4 *The PROP presented by a symmetric monoidal theory* (Σ, E)*, denoted* $\mathtt{P}(\Sigma, E)$*, is the coequalizer of the diagram*

$$\mathtt{F}E \underset{\rho}{\overset{\lambda}{\rightrightarrows}} \mathtt{F}\Sigma.$$

The intuition is that the coequalizer is the freest PROP subject to the constraints that the 'left-hand side' of each equation (α, β), given by λ, is equal to the 'right-hand side', given by ρ.

Definition 5 *A* **subPROP** $\mathtt{P}'$ *of a given PROP* $\mathtt{P}$ *is the source of a monomorphism in* PROP*,* $i\colon \mathtt{P}' \to \mathtt{P}$*. A* **quotient PROP** $\mathtt{Q}$ *of a given PROP* $\mathtt{P}$ *is the target of a regular epimorphism in* PROP*,* $\phi\colon \mathtt{P} \to \mathtt{Q}$*.*

We are often interested in comparing PROPs that have similar generators and equations. The next proposition can be phrased as the slogan, "Adding generators and removing equations both result in bigger PROPs." Once again, a proof of this proposition will appear in [2].

Proposition 6 *Given a symmetric monoidal theory* (Σ, E)*, a signature* Σ' *such that* $\Sigma \subseteq \Sigma'$*, and equations* $E' \subseteq E$*, the following are true:*

- $\mathtt{P}(\Sigma, E)$ *is a subPROP of* $\mathtt{P}(\Sigma', E)$*, and*
- $\mathtt{P}(\Sigma, E)$ *is a quotient PROP of* $\mathtt{P}(\Sigma, E')$*.*

It immediately follows that $\mathtt{P}(\Sigma', E)$ is a quotient PROP of, and $\mathtt{P}(\Sigma, E')$ is a subPROP of $\mathtt{P}(\Sigma', E')$. Another immediate corollary is that $\mathtt{P}(\Sigma, E)$ is a quotient PROP of the free PROP $\mathtt{F}\Sigma$.

In later chapters we will show a PROP is the PROP for a symmetric monoidal theory (Σ, E) by finding 'standard forms' for the morphisms.

Definition 7 *Given a symmetric monoidal theory* (Σ, E) *and a PROP* $\mathtt{P}$ *such that* $\phi\colon \mathtt{F}\Sigma \to \mathtt{P}$ *is an epimorphism in* PROP*, a* **standard form** *for a morphism* p *in* $\mathtt{P}$ *is a particular morphism* $\tilde{p}$ *in* $\mathtt{F}\Sigma$ *such that* $\phi\tilde{p} = p$*.*

The requirement that ϕ is an epimorphism in PROP means ϕ is surjective on morphisms. Thus every morphism in $\mathtt{P}$ has a standard form. There is no requirement that standard forms respect composition, so we do not get a functor $\mathtt{P} \to \mathtt{F}\Sigma$ that satisfies $p \mapsto \tilde{p}$. However, there is a functor $\nu\colon \mathtt{UP} \to \mathtt{UF}\Sigma$ satisfying $\nu(\mathtt{U}p) = \mathtt{U}\tilde{p}$ since signatures are discrete.

Proposition 8 *Given a PROP* $\mathtt{P}$ *and a symmetric monoidal theory* (Σ, E)*, let* $\pi\colon \mathtt{F}\Sigma \to \mathtt{Q}$ *be the coequalizer of* $\mathtt{F}\mathcal{E} \rightrightarrows \mathtt{F}\Sigma$*. Given any epimorphism* $\phi\colon \mathtt{F}\Sigma \to \mathtt{P}$*, such that* $\phi\lambda = \phi\rho$*, and* $\pi(f) = \pi\widetilde{\phi(f)}$ *for all morphisms* f *in* $\mathtt{F}\Sigma$*, then* $\mathtt{P}$ *and* $\mathtt{Q}$ *are isomorphic PROPs.*

This theorem says that if $\mathtt{P}$ 'respects the equations' of the symmetric monoidal theory and every morphism in the free PROP can be connected to a standard form using the equations in E, then $\mathtt{P}$ is $\mathtt{P}(\Sigma, E)$.

Proof. As noted above, there is a functor $\nu\colon \mathtt{UP} \to \mathtt{UF}\Sigma$ that sends morphisms in $\mathtt{P}$ to their standard forms, on the level of signatures. That is, $\nu f = \tilde{f}$. The condition $\phi\lambda = \phi\rho$ means there is a unique morphism $\alpha\colon \mathtt{Q} \to \mathtt{P}$ such that $\phi = \alpha\pi$. We will show α is an isomorphism by showing $\mathtt{U}\alpha$ is an isomorphism and lifting this isomorphism of signatures to an isomorphism of PROPs. It is immediately evident that $\mathtt{U}\phi \circ \nu = 1_{\mathtt{UP}}$, since the image of the standard form of a morphism is the same as the original morphism. Thus $\mathtt{U}\alpha \circ \mathtt{U}\pi \circ \nu = 1$. It remains to show $\mathtt{U}\pi \circ \nu$ is a two-sided inverse.

Since $\pi(f) = \pi\widetilde{\phi(f)}$, applying the functor $\mathtt{U}$ gives $\mathtt{U}\pi(f) = \mathtt{U}\pi \circ \widetilde{\mathtt{U}\phi(f)} = \mathtt{U}\pi \circ \nu(\mathtt{U}\phi(f))$. Thus $\mathtt{U}\pi = \mathtt{U}\pi \circ \nu \circ \mathtt{U}\alpha \circ \mathtt{U}\pi$. Now π is a regular epimorphism and $\mathtt{U}$ is a monadic functor over $\mathrm{Set}^{\mathbb{N}\times\mathbb{N}}$, a topos in which epimorphisms split (*i.e.* the Axiom of Choice holds), so $\mathtt{U}\pi$ is an epimorphism [8, Thm. 4.4.4]. This means $\mathtt{U}\alpha$ can be cancelled on the right, giving $1 = \mathtt{U}\pi \circ \nu \circ \mathtt{U}\alpha$. This shows $\mathtt{U}\pi \circ \nu$ is a two-sided inverse to $\mathtt{U}\alpha$. Because $\mathtt{U}$ is monadic, $\mathtt{U}$ reflects isomorphisms [8, *loc. cit.*], which means α is an isomorphism, so $\mathtt{P} \cong \mathtt{Q}$. ■

Chapter 3

Generators and equations description of $\mathtt{FinRel}_k$

Now that we have the proper tools for presenting PROPs in terms of generators and equations, we turn our attention to the PROP $\mathtt{FinRel}_k$, which we will use as the target for several PROP morphisms. In what follows we fix a field k, and all vector spaces will be over this field.

Definition 9 *Given vector spaces U and V, a* **linear relation** *$L\colon U \nrightarrow V$, is a linear subspace*

$$L \subseteq U \oplus V.$$

In particular, a linear relation $L\colon k^m \nrightarrow k^n$ is just an arbitrary system of linear equations relating m input variables to n output variables. This is why linear relations are fundamental to control theory.

Since the direct sum $U \oplus V$ is also the cartesian product of U and V, a linear relation is indeed a relation in the usual sense, but with the property that if $u \in U$ is related to $v \in V$ and $u' \in U$ is related to $v' \in V$ then $cu + c'u'$ is related to $cv + c'v'$

whenever $c, c' \in k$. We compose linear relations $L\colon U \nrightarrow V$ and $L'\colon V \nrightarrow W$ in the usual way of composing relations:

$$L'L = \{(u,w)\colon\ \exists\, v \in V \ \ (u,v) \in L \text{ and } (v,w) \in L'\}.$$

There is thus a category FinRel_k whose objects are finite-dimensional vector spaces over k, and whose morphisms are linear relations.

Moreover, FinRel_k becomes symmetric monoidal, with the direct sum of vector spaces providing the symmetric monoidal structure. In particular, given linear relations $L\colon U \nrightarrow V$ and $L'\colon U' \nrightarrow V'$, the linear relation $L \oplus L'\colon U \oplus U' \nrightarrow V \oplus V'$, is given by

$$L \oplus L' = \{(u,u',v,v')\colon\ (u,v) \in L \text{ and } (u',v') \in L'\}.$$

Any linear map $f\colon U \to V$ gives a linear relation $F\colon U \nrightarrow V$, namely the graph of that map:

$$F = \{(u, f(u))\colon u \in U\}.$$

Composing linear maps thus becomes a special case of composing linear relations. Thus, the category $\mathrm{FinVect}_k$ of finite-dimensional vector spaces and linear maps is a subcategory of FinRel_k. If we make $\mathrm{FinVect}_k$ into a symmetric monoidal category using direct sum, the inclusion of $\mathrm{FinVect}_k$ in FinRel_k is a symmetric monoidal functor.

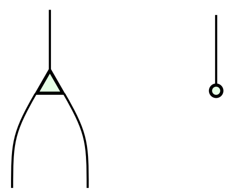
To work with FinRel_k using the machinery of PROPs, we make the following definition:

Definition 10 *For any field k, let $\mathtt{FinRel}_k$ be the PROP where a morphism from m to n is a linear relation from k^m to k^n, with the usual composition of relations, with direct sum providing the tensor product.*

One can check that $\mathtt{FinRel}_k$ is equivalent, as a symmetric monoidal category, to FinRel_k. It is a skeleton of FinRel_k, so it is clearly equivalent as a category. However, note that $\mathtt{FinRel}_k$ has trivial associators and unitors (being a PROP), while FinRel_k does not, so the inclusion of $\mathtt{FinRel}_k$ in FinRel_k is not a *strict* symmetric monoidal functor.

Our generators for $\mathtt{FinRel}_k$ are logically organized into three pairs together with one 'scaling' morphism for each element of k. We make use of string diagrams to elucidate various compositions.

The first pair is duplication and deletion:



.

Duplication is the linear relation $\Delta\colon k \nrightarrow k^2$ given by

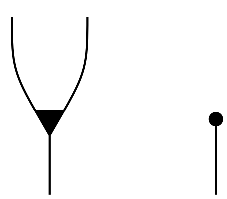
$$\Delta = \{(x, x, x) : x \in k\} \subseteq k \oplus k^2.$$

That is, Δ outputs two copies of its input. Deletion is the linear relation $!\colon k \nrightarrow \{0\}$ given by

$$! = \{(x, 0) : x \in k\} \subseteq k \oplus \{0\},$$

where $\{0\}$ is the zero-dimensional vector space. Thus ! 'eats up' its input, yielding no output. Both of these linear relations are also linear maps: Δ is the diagonal map, while ! is the unique (linear) map to $\{0\}$.

Our next pair is addition and zero:



.

Addition is the linear relation $+\colon k^2 \nrightarrow k$ given by

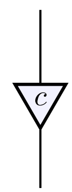
$$+ = \{(x, y, x + y) : x, y \in k\} \subseteq k^2 \oplus k.$$

That is, its output is the sum of its two inputs. Zero is the linear relation $0\colon \{0\} \nrightarrow k$ given by

$$0 = \{(0,0)\} \subseteq \{0\} \oplus k.$$

Thus 0 takes no input and outputs the *number* 0. As with the first pair, these linear relations are also linear maps, where 0 is the unique linear map from $\{0\}$.

For any $c \in k$, the scaling morphism s_c, depicted



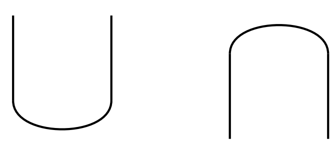
,

is the linear relation $s_c\colon k \nrightarrow k$ given by

$$s_c = \{(x, cx) : x \in k\} \subseteq k \oplus k.$$

Thus s_c scales its input by a factor of c. That each s_c is a linear map is a direct consequence of the closure of multiplication in any field.

The final pair is cup and cap:



.

Cup is the linear relation $\cup\colon k^2 \nrightarrow \{0\}$ given by

$$\cup = \{(x, x, 0) : x \in k\} \subseteq k^2 \oplus \{0\}.$$

Thus $\cup$ is a partial function and not a linear map. Cap is the linear relation $\cap\colon \{0\} \nrightarrow k^2$ given by

$$\cap = \{(0, x, x) : x \in k\} \subseteq \{0\} \oplus k^2.$$

Thus $\cap$ is a multi-valued function and not a linear map. Informally, both $\cup$ and $\cap$ can be thought of as 'bent identity morphisms', where the two inputs (*resp.* outputs) are identified. Bending a string twice allows the output of a morphism to affect its own input, which gives us a way to model feedback in a control system.

While other choices can be made for the generators, this choice has the advantage that all the generators are linear maps, with the exception of $\cup$ and $\cap$. Omitting these generators and the equations that include them leaves us with a presentation for the subPROP $\mathtt{FinVect}_k \subseteq \mathtt{FinRel}_k$ of finite-dimensional vector spaces over k and linear maps, which is another important category.

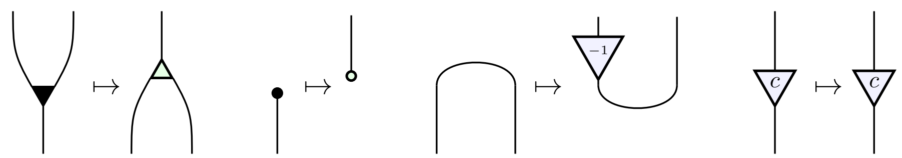
While the list of equations in our presentation of $\mathtt{FinRel}_k$ is lengthy, they can be summarized as those necessary for several nice properties to hold:

1. $(k, +, 0, \Delta, !)$ is a bicommutative bimonoid;
2. $+$, 0, Δ and $!$ commute with scaling;
3. $\cup$ and $\cap$ obey the zigzag equations;
4. $(k, +, 0, +^\dagger, 0^\dagger)$ is a commutative extra-special $\dagger$-Frobenius monoid;
5. $(k, \Delta^\dagger, !^\dagger, \Delta, !)$ is a commutative extra-special $\dagger$-Frobenius monoid;
6. the field operations of k can be recovered from the generators.

Note that item (3) makes $\mathtt{FinRel}_k$ into a $\dagger$-compact category, allowing us to mention the adjoints of generating morphisms in the subsequent properties.

The word 'separable' is sometimes used as a synonym for 'special' here [27]. A Frobenius monoid is **special** if the comultiplication followed by the multiplication is equal to the identity. An **extra-special** Frobenius monoid has an additional, less common property: the unit followed by the counit of the monoid is also equal to the identity (but now the identity on the unit object for the tensor product). This 'extra' equation is one of two from the four bimonoid equations that can be added to a †-Frobenius monoid without making the monoid trivial [15]. The other bimonoid equation that can be added is a consequence of the 'special' equation. See Appendix A.5 for a demonstration. Because of its graphical depiction, the extra equation has been called the 'bone' equation by others [13, 31].

We have placed some emphasis on the fact that cup and cap obey the zigzag equations, which allows for a duality functor, $\dagger\colon \mathtt{FinRel}_k \to \mathtt{FinRel}_k$, which 'turns morphisms around'. There is another duality on $\mathtt{FinRel}_k$ that is somewhat subtler. The functor $*\colon \mathtt{FinRel}_k \to \mathtt{FinRel}_k$ 'turns morphisms around' *and* 'swaps the color' of morphisms. To wit, $+^* = \Delta$, $0^* = !$, $\cap^* = (-1 \oplus 1) \circ \cup$, and $s_c^* = s_c$:



The extra factor of -1 in $\cap^*$ may seem surprising, given that cap and cup do not appear to have any colors to swap, and turning the morphism around just alternates between cap and cup. We shall later see equations **(29)** and **(30)**, which show the cap and cup do have an implicit color that is swapped here. As with $\dagger$, $f^{**} = f$ for any morphism f. Other authors, such as Sobociński [29], have compared this second duality to the Bizarro World in the Superman universe, where 'good' and 'evil' are swapped, leading them to refer to this duality as 'bizarro' duality. Unlike the $-^\dagger$ duality, the $-^*$ duality can be restricted to a duality on $\mathtt{FinVect}_k$, in which case it is identifiable with transposition.[1]

3.1 Presenting $\mathtt{FinVect}_k$

As a warmup for our presentation of $\mathtt{FinRel}_k$, in this section we give a presentation of a simpler PROP called $\mathtt{FinVect}_k$, in which the morphisms are linear *maps*, rather than

[1]It is also possible to encode complex numbers so that f^* is the *conjugate* transpose of f, as in [30].

fully general linear *relations*. Our generators for $\mathtt{FinVect}_k$ are a subset of our generators for $\mathtt{FinRel}_k$: we simply leave out the cup and cap, and keep the rest. The equations amount to saying:

1. $(k, +, 0, \Delta, !)$ is a bicommutative bimonoid;
2. $+$, 0, Δ and $!$ commute with scaling;
3. the rig operations of k can be recovered from the generators.

Here a **rig** is a 'ring without negatives', so the rig operations of k are the binary operations of addition and multiplication, together with the nullary operations (or constants) 0 and 1.

In Definition 10, we said that in the PROP $\mathtt{FinRel}_k$ the morphisms from m to n are linear relations $L\colon k^m \nrightarrow k^n$. We have seen that linear maps are a special case of linear relations. Thus we make the following definition:

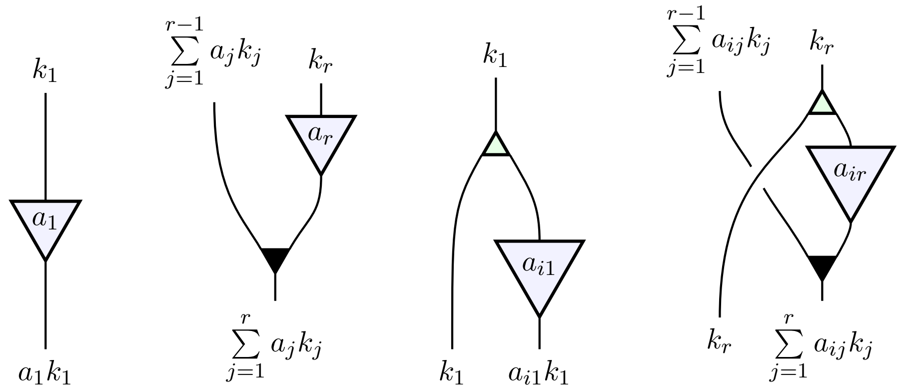
Definition 11 *Let* $\mathtt{FinVect}_k$ *be the subPROP of* $\mathtt{FinRel}_k$ *whose morphisms are linear maps.*

One can check that $\mathtt{FinVect}_k$ is equivalent as a symmetric monoidal category to $\mathrm{FinVect}_k$, where the objects are *all* finite-dimensional vector spaces over k and where the morphisms are linear maps between these.

Lemma 12 *For any field k, the PROP* $\mathtt{FinVect}_k$ *is generated by these morphisms:*

1. *scaling* $s_c\colon k \to k$ *for any* $c \in k$
2. *addition* $+\colon k \oplus k \to k$
3. *zero* $0\colon \{0\} \to k$
4. *duplication* $\Delta\colon k \to k \oplus k$
5. *deletion* $!\colon k \to \{0\}$

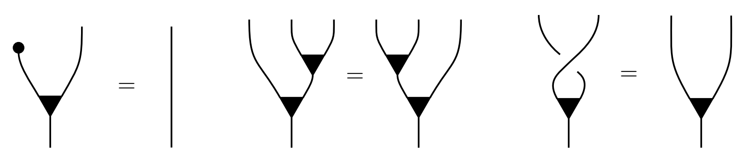
Proof. By this we mean that every morphism in $\mathtt{FinVect}_k$ can be obtained from these morphisms using composition, tensor product, identity morphisms and the braiding. A linear map in $\mathtt{FinVect}_k$, $T\colon k^m \to k^n$ can be expressed as n k-linear combinations of m elements of k. That is, $T(k_1, \ldots, k_m) = (\sum_j a_{1j}k_j, \ldots, \sum_j a_{nj}k_j)$, $a_{ij} \in k$. Any k-linear combination of r elements can be constructed with only addition, multiplication, and zero, with zero only necessary when providing the unique k-linear combination for $r = 0$. When $r = 1$, $a_1(k_1)$ is an arbitrary k-linear combination. For $r > 1$, $+(S_{r-1}, a_r(k_r))$ yields an arbitrary k-linear combination on r elements, where S_{r-1} is an arbitrary k-linear combination of $r-1$ elements. The inclusion of duplication allows the process of forming k-linear combinations to be repeated an arbitrary (finite) positive number of times, and deletion allows the process to be repeated zero times. When n k-linear combinations are needed, each input may be duplicated $n-1$ times. Because $\mathtt{FinVect}_k$ is being generated as a PROP, the mn outputs can then be permuted into n collections of m outputs: one output from each input for each collection. Each collection can then form a k-linear combination, as above. The following diagrams illustrate the pieces that form this inductive argument.



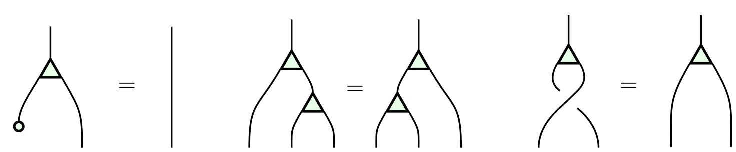
Since scaling provides the map $k_1 \mapsto a_1k_1$, as in the far left diagram, the middle-left diagram can be used inductively to form a k-linear combination of any number of inputs. In particular, we have any linear map $S_r\colon k^m \to k$ given by $(k_1, \ldots, k_m) \mapsto (\sum_j a_{rj}k_j)$. Using duplication as in the middle-right diagram, one can produce the map $k_1 \mapsto (k_1, a_{i1}k_1)$, to which the right diagram can be inductively applied. Thus we can build any linear map, $T_j \in \texttt{FinVect}_k$, $T_j \colon k^m \to k^{m+1}$ given by $(k_1, \ldots, k_m) \mapsto (k_1, \ldots, k_m, \sum_j a_{ij} k_j)$. If we represent the identity map on k^r as 1^r, the r-fold monoidal product of the identity map on k, any linear map $T \colon k^m \to k^n$ can be given by $(k_1, \ldots, k_m) \mapsto (\sum_j a_{1j} k_j, \ldots, \sum_j a_{nj} k_j)$, which can be expressed as $T = (S_1 \oplus 1^{n-1})(T_2 \oplus 1^{n-2}) \cdots (T_{n-1} \oplus 1^1) T_n$. The above works as long as $m, n \neq 0$. Otherwise, $f \colon k^m \to \{0\}$ can be written as an m-fold tensor product of deletion, $!^m$, and $f \colon \{0\} \to k^n$ can be written as an n-fold tensor product of zero, 0^n. Since $f \colon \{0\} \to \{0\}$ is the monoidal unit, this has an empty diagram for its string diagram. ■

It is easy to see that the morphisms given in Lemma 12 obey the following 18 equations:

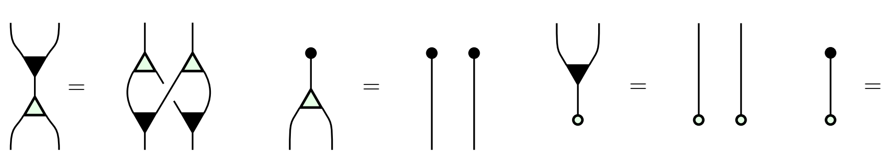
(1)–(3) Addition and zero make k into a commutative monoid:



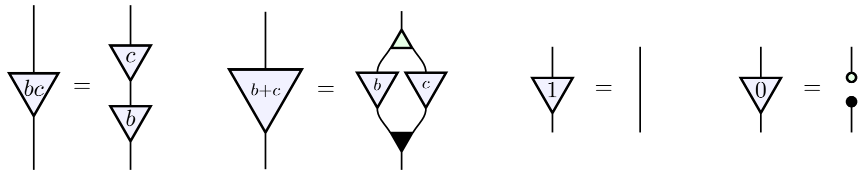
(4)–(6) Duplication and deletion make k into a cocommutative comonoid:



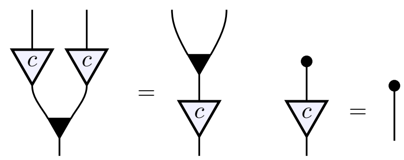
(7)–(10) The monoid and comonoid structures on k fit together to form a bimonoid:



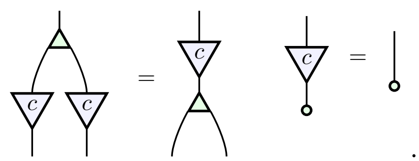
(11)–(14) The rig structure of k can be recovered from the generating morphisms:



(15)–(16) Scaling commutes with addition and zero:



(17)–(18) Scaling commutes with duplication and deletion:



In fact, these equations are enough: any two ways of drawing a linear map as a signal-flow diagram can be connected using these equations. That is, together with the generators, they give a presentation of $\mathtt{FinVect}_k$ as a PROP. More precisely, the generating morphisms are the $\mathtt{F}$-images of a signature $\Sigma_{\mathtt{FinVect}_k}$, these 18 equations are the $2|k|^2 + 4|k| + 12$ pairs[2] of Σ-terms $E_{\mathtt{FinVect}_k}$, and $\mathtt{FinVect}_k$ is the coequalizer of $\mathtt{F}E_{\mathtt{FinVect}_k} \rightrightarrows \mathtt{F}\Sigma_{\mathtt{FinVect}_k}$. Thus $\mathtt{FinVect}_k$ is the PROP presented by the symmetric monoidal theory $(\Sigma_{\mathtt{FinVect}_k}, E_{\mathtt{FinVect}_k})$.

Theorem 13 *The PROP* $\mathtt{FinVect}_k$ *is presented by the generators given in Lemma 12, and equations* **(1)–(18)** *as listed above.*

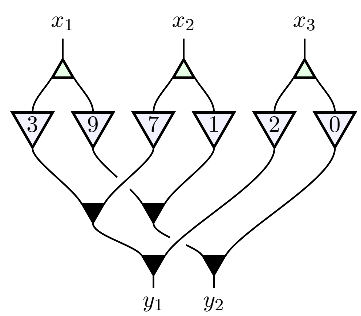
[2]Equations **(11)** and **(12)** listed above are each $|k|^2$ pairs of Σ-terms, and equations **(15)–(18)** are each $|k|$ pairs of Σ-terms.

Proof. To prove this, we find a standard form for morphisms in $\mathtt{FinVect}_k$ and use Theorem 8. That is, it suffices to show that any string diagram built from generating morphisms and the braiding can be put into a standard form using topological equivalences and equations **(1)–(18)**.

A qualitative description of this standard form will be helpful for understanding how an arbitrary string diagram can be rewritten in this form. By way of example, consider the linear transformation $T\colon \mathbb{R}^3 \to \mathbb{R}^2$ given by

$$(x_1, x_2, x_3) \mapsto (y_1, y_2) = (3x_1 + 7x_2 + 2x_3, 9x_1 + x_2).$$

Its standard form looks like this:



.

This is a string diagram picture of the following equation:

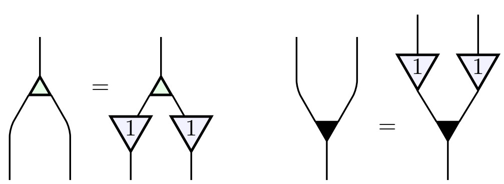
$$Tx = \begin{bmatrix} 3 & 7 & 2 \\ 9 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

In general, given a k-linear transformation $T\colon k^m \to k^n$, we can describe it using an $n \times m$ matrix with entries in k. The case where m and/or n is zero gives a matrix with no entries, so their standard form will be treated separately. For positive values of m and n, the standard form has three distinct layers. The top layer consists of m clusters of $n-1$ instances of Δ. The middle layer is mn scalings. The n outputs of the jth cluster connect to the inputs of the scalings by $\{a_{1j}, \ldots, a_{nj}\}$, where a_{ij} is the ij entry of A, the matrix for

T. The bottom layer consists of n clusters of $m-1$ instances of $+$. There will generally be braiding in this layer as well, but since the category is being generated as symmetric monoidal, the locations of the braidings doesn't matter so long as the topology of the string diagram is preserved. The topology of the sum layer is that the ith sum cluster gets its m inputs from the outputs of the scalings by $\{a_{i1}, \ldots, a_{im}\}$. The arrangement of the instances of Δ and $+$ within their respective clusters does not matter, due to the associativity of $+$ via equation **(2)** and coassociativity of Δ via equation **(5)**. For the sake of making the standard form explicit with respect to these equations, we may assume the right output of a Δ is always connected to a scaling input, and the right input of a $+$ is always connected to a scaling output. This gives a prescription for drawing the standard form of a string diagram with a corresponding matrix A.

The standard form for $T\colon k^0 \to k^n$ is n zeros $(0 \oplus \cdots \oplus 0)$, and the standard form for $T\colon k^m \to k^0$ is m deletions $(! \oplus \cdots \oplus !)$.

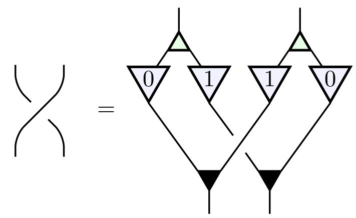
Each of the generating morphisms can easily be put into standard form: the string diagrams for zero, deletion, and scaling are already in standard form. The string diagram for duplication (*resp.* addition) can be put into standard form by attaching the scaling s_1, equation **(13)**, to each of the outputs (*resp.* inputs).



The braiding morphism is just as basic to our argument as the generating morphisms, so we will need to write the string diagram for B in standard form as well. The matrix corresponding to braiding is

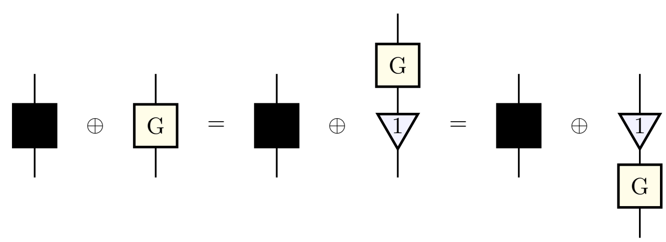
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

so its standard form is as follows:



For $n > 1$, any morphism built from n copies of the **basic morphisms**—that is, generating morphisms and the braiding—can be built up from a morphism built from $n-1$ copies by composing or tensoring with one more basic morphism. Thus, to prove that any string diagram built from basic morphisms can be put into its standard form, we can proceed by induction on the number of basic morphisms.

Furthermore, because strings can be extended using the identity morphism, equation **(13)** can be used to show tensoring with any generating morphism is equivalent to tensoring with 1, followed by a composition: $\Delta = \Delta \circ 1$, $+ = 1 \circ +$, $c = 1 \circ c$, $! =! \circ 1$, $0 = 1 \circ 0$. In the case of braiding, the step of tensoring with 1 is repeated once before making the composition: $B = (1 \oplus 1) \circ B$.



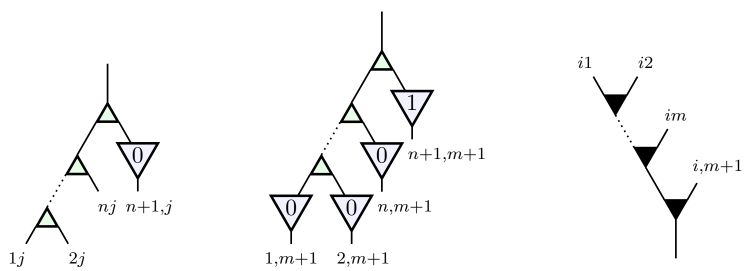
Thus there are 11 cases to consider for this induction: $\oplus 1$, $+\circ$, $\circ\Delta$, $\Delta\circ$, $\circ+$, $\circ c$, $c\circ$, $\circ 0$, $!\circ$, $B\circ$, $\circ B$. Without loss of generality, the string diagram S to which a generating morphism is added will be assumed to be in standard form already. Labels ij on diagrams illustrating these cases correspond to strings incident to the scalings by a_{ij}.

- **$\oplus$1**

When tensoring morphisms together, the matrix corresponding to $C \oplus D$ is the block diagonal matrix

$$\begin{bmatrix} C & 0 \\ 0 & D \end{bmatrix},$$

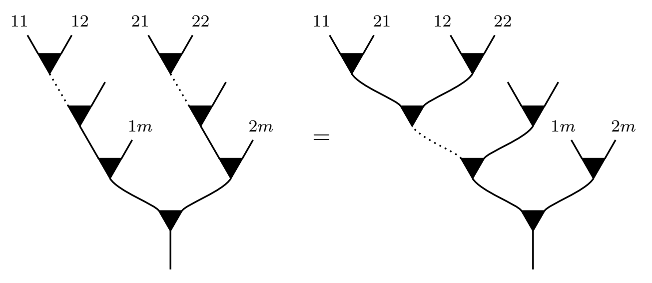
where, by abuse of notation, the block C is the matrix corresponding to morphism C, and respectively D with D. Thus, when tensoring S by 1, we write the matrix for S with one extra row and one extra column. Each of these new entries will be 0 with the exception of a 1 at the bottom of the extra column. The string diagram corresponding to the new matrix can be drawn in standard form as prescribed above. Using equations **(14)**, **(4)**, and **(1)**, the standard form reduces to $S \oplus 1$. The process is reversible, so if the string diagram S can be drawn in standard form, the string diagram $S \oplus 1$ can be drawn in standard form, too. The diagrams below show the relevant strings before they are reduced.



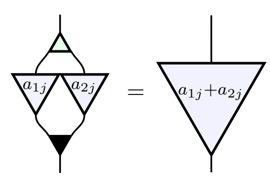
Note that for $i = n + 1$, $a_{i2} = \cdots = a_{im} = 0$, so the scalings going to the sum cluster will be s_0, and $a_{i,m+1} = 1$. Otherwise $a_{i,m+1} = 0$, and the rest of the a_{ij} depend on the matrix corresponding to S. When $S = (! \oplus \cdots \oplus !)$, the matrix corresponding to $S \oplus 1$ has a single row, $(0 \cdots 0\ 1)$, and the standard form generated is just the middle diagram above. When the same simplifications are applied, no sum cluster exists to eliminate the zeros, so the standard form still simplifies to $S \oplus 1$. Dually, when $S = (0 \oplus \cdots \oplus 0)$, the matrix representation of $S \oplus 1$ is a column matrix. No duplication cluster exists in the standard form for this matrix, so the same simplifications again reduce to $S \oplus 1$.

- $+\circ$

  If we compose the string diagram for addition with S, first consider only the affected clusters of additions: two clusters are combined into a larger cluster. Without loss of generality we can assume these are the first two clusters, or formally, $(+ \oplus 1^{n-2})(S)$. We can rearrange the sums using the associative law, equation **(2)**, and permute the inputs of this large cluster using the commutative law, equation **(3)**. After several iterations of these two equations, the desired result is obtained:

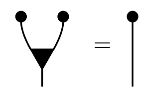


  Now the right side of equation **(12)** appears in the diagram m times with a_{1j} and a_{2j} in place of b and c. Equation **(12)** can therefore be used to simplify to the scalings of $a_{1j} + a_{2j}$.



  The simplification removes one instance of Δ from each of the m clusters of Δ and m instances of $+$ from the large addition cluster. There will remain $(m-1)+(m-1)+(1)-(m) = m-1$ instances of $+$, which is the correct number for the cluster. *I.e.* the composition has been reduced to standard form.

  The argument is vastly simpler if $S = (0 \oplus \cdots \oplus 0)$. In that case equation **(1)** deletes the addition and one of the 0 morphisms, and S is still in the same form.

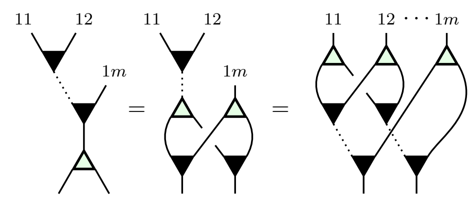


- $\circ\boldsymbol{\Delta}$

  The argument for $S \circ (\Delta \oplus 1^{m-2})$ is dual to the above argument, using the light equations **(4)**, **(5)** and **(6)** instead of the dark equations **(1)**, **(2)** and **(3)**.

- $\boldsymbol{\Delta}\circ$

  For $(\Delta \oplus 1^{n-1}) \circ S$, equation **(7)** can be used iteratively to 'float' the Δ layer above each of the two $+$ clusters formed by the first iteration.



  Each of these instances of Δ can pass through the scaling layer to Δ clusters using equation **(17)**.

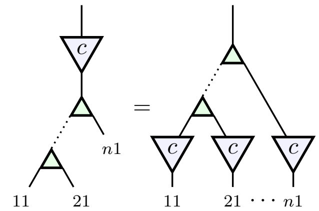
  As before, we consider the subcase $S = (0 \oplus \cdots \oplus 0)$ separately. Equation **(8)** removes the duplication and creates a new zero, so S remains in the same form.

- $\circ+$

  For $S(+ \oplus 1^{m-1})$, the argument is dual to the previous one: equation **(7)** is used to 'float' the additions down, equation **(15)** sends the additions through the scalings, and equation **(9)** removes the addition and creates a new deletion in the subcase $S = (! \oplus \cdots \oplus !)$.

- $\circ\boldsymbol{s_c}$

  We can iterate equation **(17)** when a scaling is composed on top, as in $S(s_c \oplus 1^{m-1})$.



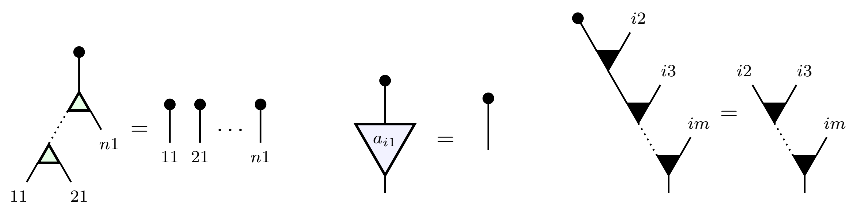
The double scalings in the scaling layer reduce to a single scaling via equation **(11)**, $s_c \circ a_{ij} = ca_{ij}$, which leaves the diagram in standard form. The composition does nothing when $S = (! \oplus \cdots \oplus !)$, due to equation **(18)**.

- $\boldsymbol{s_c \circ}$

  A dual argument can be made for $(s_c \oplus 1^{n-1}) \circ S$ using equations **(15)**, **(11)** and **(16)**.

- $\circ\mathbf{0}$

  For $S(0 \oplus 1^{m-1})$, equations **(8)** and **(16)** eradicate the first Δ cluster and all the scalings incident to it, leaving behind n zeros. Equation **(1)** erases each of these zeros along with one addition per addition cluster, leaving a diagram that is in standard form.



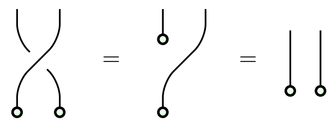
  When $S = (! \oplus \cdots \oplus !)$, the zero annihilates one of the deletions via equation **(10)**.

- $\mathbf{!}\circ$

  A dual argument erases the indicated output for the composition $(! \oplus 1^{n-1}) \circ S$ using equations **(9)**, **(18)**, and **(4)**. Again, equation **(10)** annihilates the deletion and one of the zeros if $S = (0 \oplus \cdots \oplus 0)$.

- $\boldsymbol{B}\circ$

  Since this category of string diagrams is symmetric monoidal, an appended braiding will naturally commute with the addition cluster morphisms. The principle that only the topology matters means the composition $(B \oplus 1^{n-2}) \circ S$ is in standard form. Braiding will similarly commute with deletion morphisms.

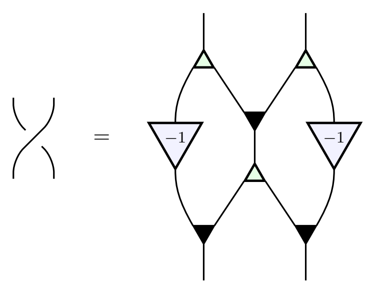


- $\circ \boldsymbol{B}$

  Composing with B on the top, braiding commutes with duplication, scaling and zero, so $S \circ (B \oplus 1^{m-2})$ almost trivially comes into standard form.

Having exhausted the ways basic morphisms can be attached to a given morphism, this completes the induction. ■

An interesting exercise is to use these equations to derive an equation that expresses the braiding in terms of other basic morphisms. One example of such a relation appeared in the introduction, Section 1.2. Here is another:



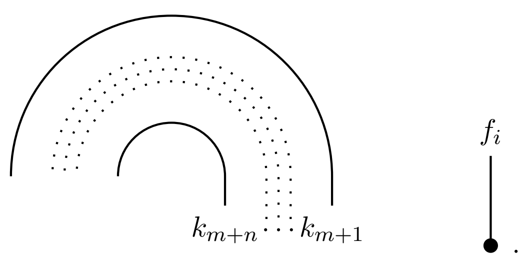
With a few more equations, $\mathrm{FinVect}_k$ can be presented as merely a monoidal category. Lafont [23] mentioned this fact, and gave a full proof in the special case where k is the field with two elements.

3.2 Presenting $\texttt{FinRel}_k$

Lemma 14 *For any field k, the PROP* $\texttt{FinRel}_k$ *is generated by these morphisms:*

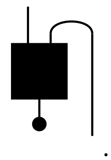
- *addition* $+\colon k \oplus k \nrightarrow k$
- *zero* $0\colon \{0\} \nrightarrow k$
- *duplication* $\Delta\colon k \nrightarrow k \oplus k$
- *deletion* $!\colon k \nrightarrow \{0\}$
- *scaling* $s_c\colon k \nrightarrow k$ *for any* $c \in k$
- *cup* $\cup\colon k \oplus k \nrightarrow \{0\}$
- *cap* $\cap\colon \{0\} \nrightarrow k \oplus k$

Proof. A morphism of FinRel_k, $R\colon k^m \nrightarrow k^n$ is a subspace of $k^m \oplus k^n \cong k^{m+n}$. In $\texttt{FinRel}_k$ this isomorphism is an equality. This subspace can be expressed as a system of k-linear equations in k^{m+n}. Theorem 13 tells us any number of arbitrary k-linear combinations of the inputs may be generated. Any k-linear equation of those inputs can be formed by setting such a k-linear combination equal to zero. In particular, if caps are placed on each of the outputs to make them inputs and all the k-linear combinations are set equal to zero, any k-linear system of equations of the inputs and outputs can be formed. Expressed in terms of string diagrams,



The left diagram turns the n outputs into inputs by placing caps on all of them. The morphism zero gives the k-linear combination zero, so an arbitrary k-linear combination in k^{m+n} is set equal to zero ($f_i = 0$) via the cozero morphism. These elements can be combined with Theorem 13 to express any system of k-linear equations in k^{m+n}. ■

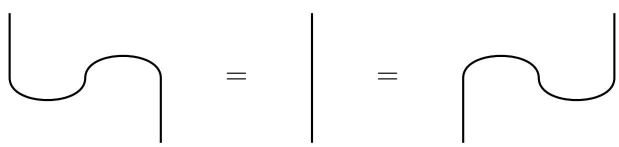
Putting these elements together, taking the $\mathtt{FinVect}_k$ portion as a black box and drawing a single string for zero or more copies of k, the picture is fairly simple:



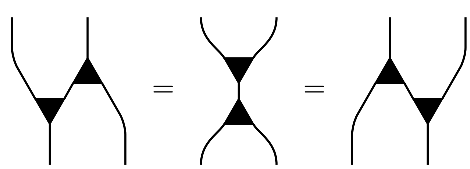
To obtain a presentation of $\mathtt{FinRel}_k$ as a PROP, we need to find enough equations obeyed by the generating morphisms listed in Lemma 14. Equations **(1)–(18)** from Theorem 13 still apply, but we need more.

For convenience, in the list below we draw the adjoint of any generating morphism by rotating it by 180°. It will follow from equations **(19)** and **(20)** that the cap is the adjoint of the cup, so this convenient trick is consistent even in that case, where *a priori* there might have been an ambiguity.

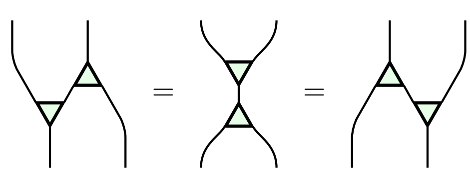
(19)–(20) $\cap$ and $\cup$ obey the zigzag equations, and thus give a †-compact category:



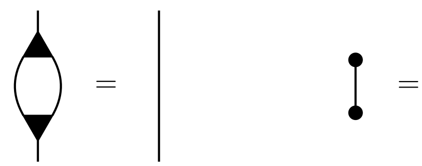
(21)–(22) $(k, +, 0, +^\dagger, 0^\dagger)$ is a Frobenius monoid:



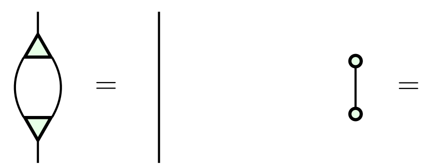
(23)–(24) $(k, \Delta^\dagger, !^\dagger, \Delta, !)$ is a Frobenius monoid:



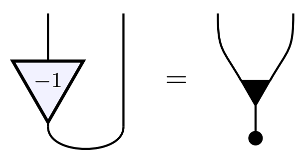
(25)–(26) The Frobenius monoid $(k, +, 0, +^\dagger, 0^\dagger)$ is extra-special:



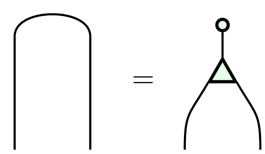
(27)–(28) The Frobenius monoid $(k, \Delta^\dagger, !^\dagger, \Delta, !)$ is extra-special:



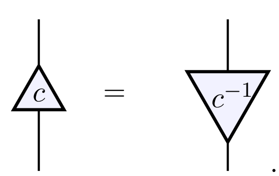
(29) $\cup$ with a scaling of -1 inserted can be expressed in terms of $+$ and 0:



(30) $\cap$ can be expressed in terms of Δ and $!$:

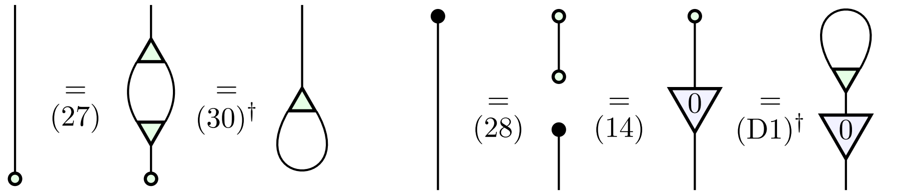


(31) For any $c \in k$ with $c \neq 0$, scaling by c^{-1} is the adjoint of scaling by c:



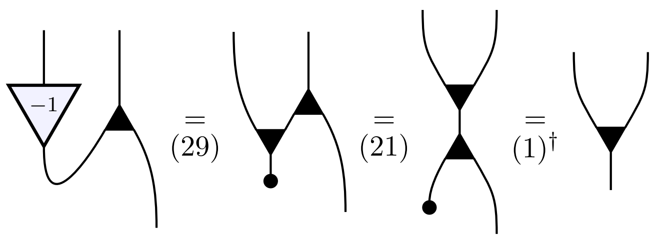
Some curious identities can be derived from equations **(1)–(31)**, beyond those already arising from **(1)–(18)**. For example:

(D1)–(D2) Deletion and zero can be expressed in terms of other generating morphisms:

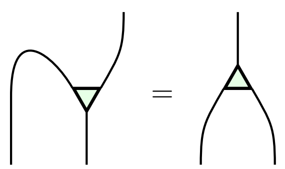


This does not diminish the role of deletion and zero. Indeed, regarding these generating morphisms as superfluous buries some of the structure of $\texttt{FinRel}_k$.

(D3) Addition can be expressed in terms of coaddition and scaling by -1, and the cup:

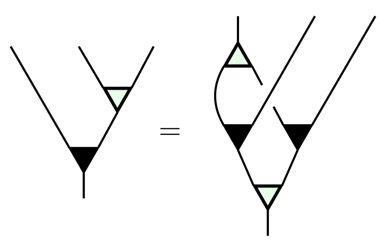


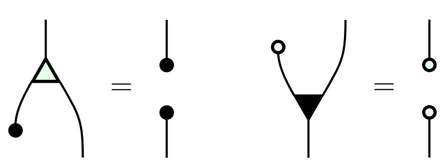
(D4) Duplication can be expressed in terms of coduplication and the cap:



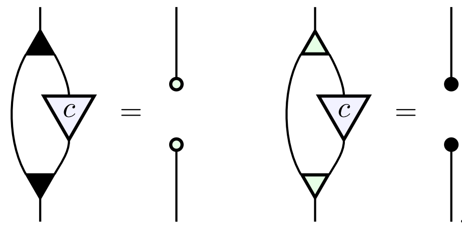
where the proof is similar to that of **(D3)**.

(D5)–(D7) We can reformulate the bimonoid equations (7)–(9) using daggers:





(D8)–(D9) When $c \neq 1$, we have:



Derived equations **(D5)–(D8)** are used below, and their proofs can be found in Appendix A. While derived equation **(D9)** is not used below, it is dual to equation **(D8)**. With a different standard form on $\texttt{FinRel}_k$, equation **(D9)** would be used in the proof of Theorem 15 below instead of equation **(D8)**.

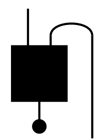
Next we show that equations **(1)–(31)** are enough to give a presentation of $\texttt{FinRel}_k$ as a PROP. In terms of symmetric monoidal theories, this means $\texttt{FinRel}_k$ is the coequalizer of $\texttt{F}E_{\texttt{FinRel}_k} \rightrightarrows \texttt{F}\Sigma_{\texttt{FinRel}_k}$, where the generating morphisms are the $\texttt{F}$-images of the signature $\Sigma_{\texttt{FinRel}_k}$, and the Σ-terms of $E_{\texttt{FinRel}_k}$ are these 31 equations. As before, we demonstrate the presentation by giving a standard form that any $\texttt{FinRel}_k$ morphism can be written in and use induction to show that an arbitrary diagram can be rewritten in its standard form using the given equations.

Theorem 15 *The PROP* $\texttt{FinRel}_k$ *is presented by the morphisms given in Lemma 14, and equations* **(1)–(31)** *as listed above.*

Proof. We prove this theorem by using the equations **(1)–(31)** to put any string diagram built from the generating morphisms and braiding into a standard form, so that any two string diagrams corresponding to the same morphism in $\texttt{FinRel}_k$ have the same standard form.

As before, we induct on the number of **basic morphisms** involved in a string diagram, where the basic morphisms are the generating morphisms together with the braiding. If we let $R\colon k^m \nrightarrow k^n$ be a morphism in $\texttt{FinRel}_k$, we can build a string diagram S for R

as in Lemma 14. Each output of S is capped, and, together with the inputs of S, form inputs for a $\texttt{FinVect}_k$ block, T. For some $r \leq m+n$, there are r outputs of T–linear combinations of the $m+n$ inputs–each set equal to zero via $(0^\dagger)^r$. When T is in standard form for $\texttt{FinVect}_k$, we say S is in **prestandard form**, and can be depicted as follows:

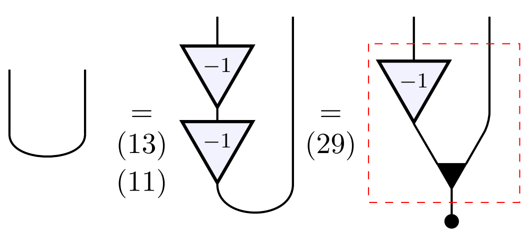


.

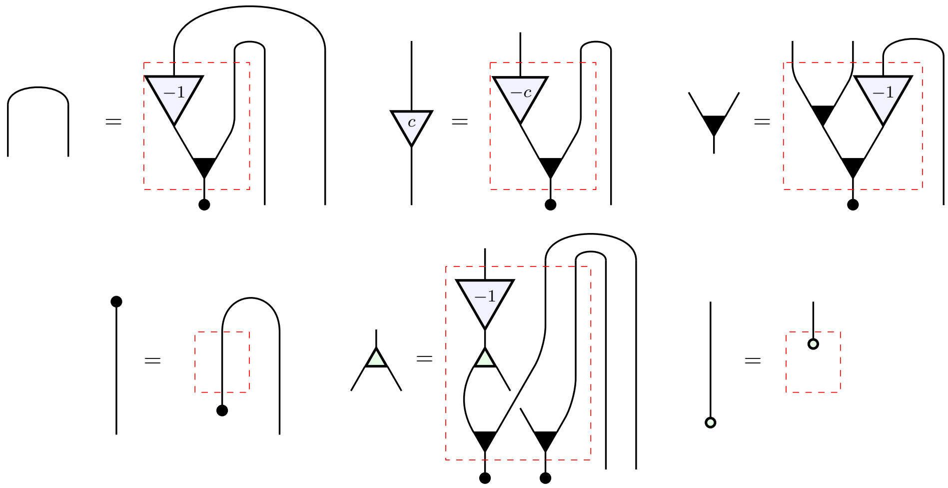
While the linear subspace of k^{m+n} defined by R is determined by a system of r linear equations, the converse is not true, meaning there may be multiple prestandard string diagrams for a single morphism R. The second stage of this proof collapses all the prestandard forms into a standard form using some basic linear algebra. The standard form will correspond to when the matrix representation of T is written in row-reduced echelon form. For this stage it will suffice to show all the elementary row operations correspond to equations that hold between diagrams. By Theorem 13, an arbitrary $\texttt{FinVect}_k$ block can be rewritten in its standard form, so the $\texttt{FinVect}_k$ blocks here need not be demonstrated in their standard form.

There are eight base cases of a string diagram with one basic morphism to consider, one case per basic morphism. In each of these basic cases, the block of the diagram equivalent to a morphism in $\texttt{FinVect}_k$ is denoted by a dashed rectangle. We first consider $\cup$.

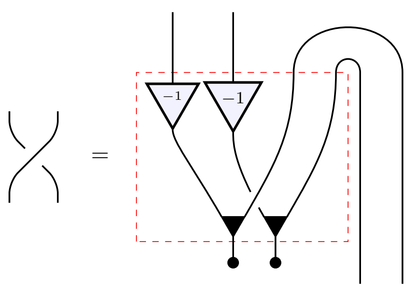
(D10)



Capping each of the inputs turns this into the standard form of $\cap$. Aside from deletion, the remaining generating morphisms can be formed by introducing a zigzag at each output and rewriting the resulting cups as above. The standard forms for 0 and ! have simpler expressions.



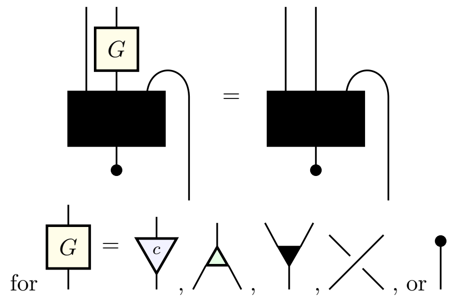
Braiding is two copies of s_1 (scaling by 1) that have been braided together.



Assuming any string diagram with j basic morphisms can be written in prestandard form, we show an arbitrary diagram with $j+1$ basic morphisms can be written in prestandard form as well. Let S be a string diagram on j basic morphisms, rewritten into prestandard form, with a maximal $\texttt{FinVect}_k$ subdiagram T. Several cases are considered: those putting a basic morphism above S, beside S, and below S.

- $\boldsymbol{S \circ G}$ **for a basic morphism** $\boldsymbol{G \neq \cap}$

  If a diagram G is composed above S, G can combine with T to make a larger $\texttt{FinVect}_k$ subdiagram if G is c, Δ, $+$, B, or 0, as these are morphisms in $\texttt{FinVect}_k$. The generating morphisms $\cap$, $\cup$ and ! are not on this list, though a composition with $\cup$ (*resp.* !) would be equivalent to tensoring by $\cup$ (*resp.* !).



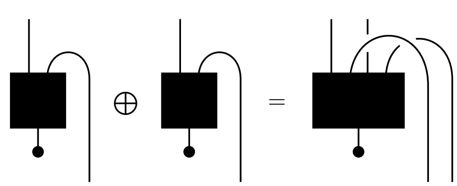
Putting these morphisms on top of S reduces to performing those compositions on T. The maximal $\texttt{FinVect}_k$ subdiagram now includes T and G, with S unchanged outside the $\texttt{FinVect}_k$ block.

- $\boldsymbol{B \circ S}$

  B commutes with caps because the category is symmetric monoidal, so capping the braiding is equivalent to putting the braiding on top of T. B is 'absorbed' into T, just as in the $S \circ G$ case.

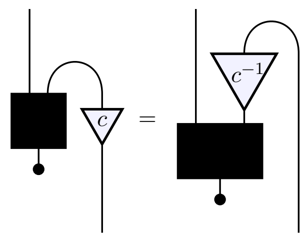
- $\boldsymbol{S \oplus G}$ **for any basic morphism** $\boldsymbol{G}$

  If any two prestandard string diagrams S and S' are tensored together, the result combines into one prestandard diagram. This is evident because the category of string diagrams is symmetric monoidal, and the $\texttt{FinVect}_k$ blocks can be placed next to each other as the tensor of two $\texttt{FinVect}_k$ blocks. These combine into a single $\texttt{FinVect}_k$ block, and absorbing all the braidings into this block as above brings the diagram into prestandard form. Since each basic morphism can be written as a prestandard diagram, the tensor $S \oplus G$ is a special case of this.



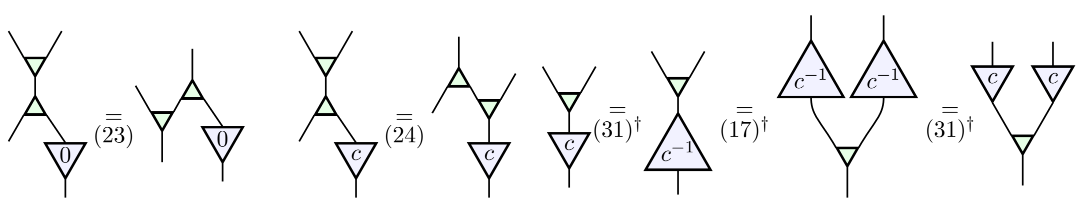
- **$s_c \circ S$ for $c \neq 0$**

  Because the outputs of S are capped, putting any morphism on the bottom of S is equivalent (via equations **(19)** and **(20)**) to putting its adjoint on top of T. Putting $c \neq 0$ below S reduces to putting c^{-1} on top of T by equation **(31)**. The case of s_0 will be considered below. The other cases of adjoints of generating morphisms that need to be considered more carefully are the ones that put $\Delta^\dagger$, $+^\dagger$ and $\cap = \cup^\dagger$ on top of T.

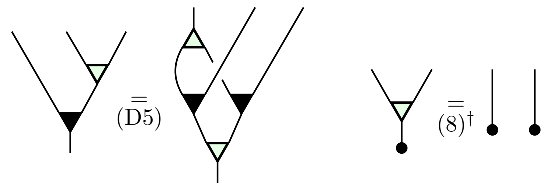


- **$\Delta \circ S$**

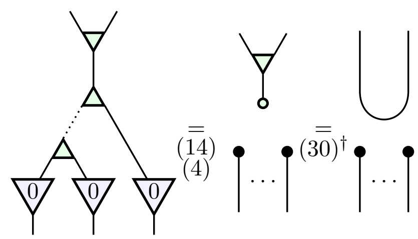
  When putting $\Delta^\dagger$ on top of T, the idea is to make it 'trickle down.' If there is a nonzero scaling incident to the Δ cluster, $\Delta^\dagger$ can slide through the Δs using equation **(23)** to the first nonzero scaling, switching to equation **(24)**. When it encounters this s_c, equation **(31)** turns c into $(c^{-1})^\dagger$, equation **(17)**† allows $\Delta^\dagger$ to pass through $(c^{-1})^\dagger$. Both copies of $(c^{-1})^\dagger$ can return to being c by another application of equation **(31)**, and the $\Delta^\dagger$ moves on to the next layer.



  When the codelta gets to a + cluster, derived equation **(D5)** has a net effect of bringing it to the bottom of the subdiagram, as the other morphisms involved all belong to $\texttt{FinVect}_k$. This allows the process to be repeated on the next addition until $\Delta^\dagger$ reaches the bottom of the + cluster. Once there, codelta interacts with the cozero layer below T; equation **(8)**† reduces it to a pair of cozeros.



If all the scalings incident to the Δ cluster are by 0, rather than trickling down, $\Delta^\dagger$ composes with ! (due to equation **(14)**), which gives $\cup$ by equation **(30)**†. By the zigzag identities, this cup becomes a cap that is tensored with a subdiagram of S that is in prestandard form.



- $+ \circ \boldsymbol{S}$

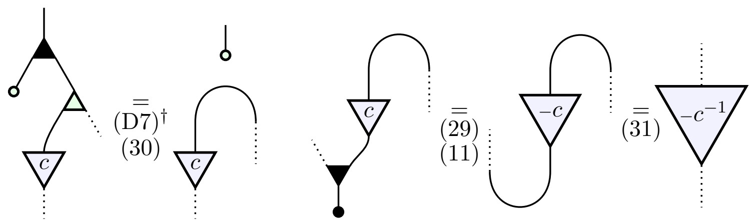
  There is a similar trickle down argument for $+^\dagger$. First rewriting all instances of s_0 via equation **(14)**, the two Δ clusters incident to the coaddition can either reduce to Δ clusters that are incident only to nonzero scalings or reduce to a single deletion, as above, if all incident scalings were s_0. There are three cases of what can happen from here.

  – **Both $\boldsymbol{\Delta}$ clusters were incident to only $\boldsymbol{s_0}$**

  In the first case, as above, the Δ clusters will reduce to ! incident to the outputs of $+^\dagger$. Equations **(D7)** and **(28)** delete the coaddition.

  – **One $\boldsymbol{\Delta}$ cluster was incident to only $\boldsymbol{s_0}$**

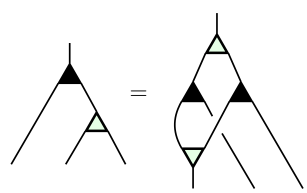
  Without loss of generality, the ! incident to $+^\dagger$ is on the left. Equation **(D7)** replaces ! and $+^\dagger$ with $!^\dagger \circ !$, and equation **(30)** replaces Δ and $!^\dagger$ with a cap. The Δ was – and the cap is – incident to some scaling s_c, $c \neq 0$. Without loss of generality, s_c is incident to the bottom addition in the cluster. Equation **(29)** replaces the addition and cozero with a cup and s_{-1}, which combines with s_c by equation **(11)**. The cup and cap turn s_{-c} around to its adjoint, which is scaling by $-c^{-1}$, by equation **(31)**.



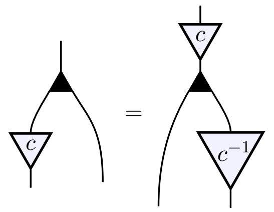
An addition cluster is above the $-c^{-1}$ scaling and a duplication cluster is below, but because those clusters are not otherwise connected to each other, there is a vertical arrangement of the morphisms in the $\texttt{FinVect}_k$ block of the string diagram such that no cups or caps are present.

– **Both Δ clusters are incident to at least one nonzero scaling**

Using equation $\mathbf{(D5)}^\dagger$, a $+^\dagger$ will pass through one Δ at a time. A new $\Delta^\dagger$ is created each time, but this can trickle down as before.

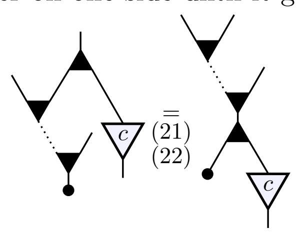


Once the $\Delta^\dagger$ trickles down, there are two possibilities for what is directly beneath each $+^\dagger$: either the same scenario will recur with a Δ connected to one or both outputs, which can only happen finitely many times, or two nonzero scalings will be below the $+^\dagger$. A scaling by any unit in k, *i.e.* $c \neq 0$, can move through a coaddition by inserting $c^{-1}c$ on the top branch and applying equation $\mathbf{(15)}^\dagger$:



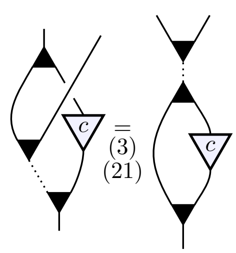
This allows one of the outputs of the coaddition to connect directly to a + cluster.

* **If both branches go to different + clusters**, Frobenius equations **(21)–(22)** slide the $+^\dagger$ down the + cluster on one side until it gets to the end of that cluster.

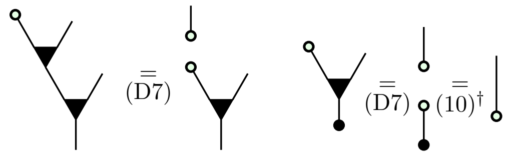


The only morphisms added to the $\mathtt{FinVect}_k$ block that are not from $\mathtt{FinVect}_k$ were the coaddition and the cozero. Since these reduce to an identity morphism string by equation $\mathbf{(1)}^\dagger$, the $\mathtt{FinVect}_k$ block is truly a $\mathtt{FinVect}_k$ block again.

* **If both branches go to the same + cluster**, equation **(3)** and the Frobenius equation **(21)** take both branches to the same addition.

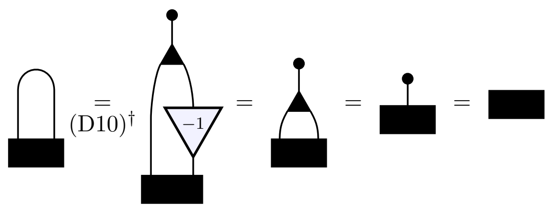


Depending on whether the remaining scaling is s_1, either equation **(25)** reduces the coaddition and the given addition to an identity string or equation **(D8)** applies. In the former case we are done, and in the latter case equations **(D7)** and $\mathbf{(10)}^\dagger$ remove the $!^\dagger$ introduced by applying equation **(D8)**.



- **$\cup \circ S$ and $S \circ \cap$**

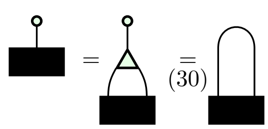
  Composing with a cup below S is equivalent to composing with cap above T, since $\cap = \cup^\dagger$. Using equation **(D10)**†, this cap can be replaced by s_{-1}, coaddition, and zero. By the arguments above, s_{-1}, $+^\dagger$, and 0 can each be absorbed into the $\texttt{FinVect}_k$ block.



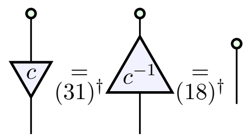
  The compositions with zero and s_{-1} expand the $\texttt{FinVect}_k$ block, thus have no effect on whether the diagram can be written in prestandard form.

- **$! \circ S$**

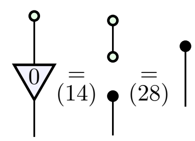
  When composing $!^\dagger$ above T, two possibilities arise, depending on whether there is a layer of Δs in the $\texttt{FinVect}_k$ block. If there is such a layer, equation **(30)** combines the $!^\dagger$ with a Δ, making a cap on top of T. As we have just seen, this can be rewritten in prestandard form.



  If no layer of Δs exists, equations **(31)**† and **(18)**† pass the codeletion through a nonzero scaling. Then equations **(D7)** and **(10)**† can be used to remove $!^\dagger$, as we have already seen. This leaves only the basic morphisms of $\texttt{FinVect}_k$ within the $\texttt{FinVect}_k$ block.

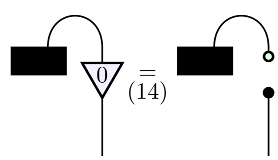


If the scaling is s_0, equation **(14)** converts s_0 to $0 \circ !$, allowing equation **(28)** to remove the $!^\dagger$, with the same conclusion.



- $\boldsymbol{s_0 \circ S}$

  Composing with s_0 below S is equivalent to composing with codeletion, followed by tensoring with zero. Codeletion is the $! \circ S$ case, and zero can be written in a prestandard form, so this reduces to tensoring two diagrams that are in prestandard form.

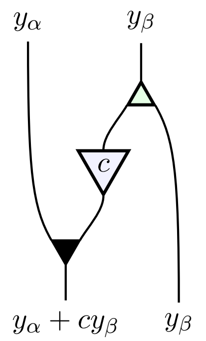


Finally, we need to show the prestandard forms can be rewritten in standard form. We need to show what elementary row operations look like in terms of string diagrams. We also need to show for an arbitrary prestandard string diagram S with $\texttt{FinVect}_k$ block T that if T is replaced with T', the diagram where an elementary row operation has been performed on T, the resulting diagram S' can be built from S using equations **(1)–(31)**.

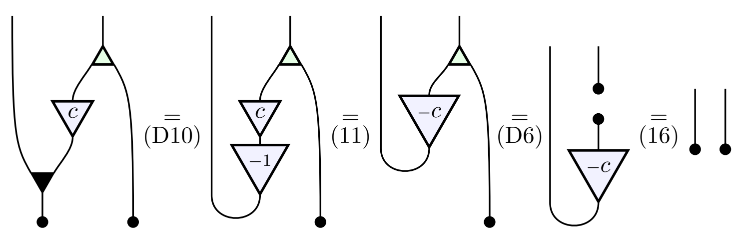
Because the ith output of a $\texttt{FinVect}_k$ diagram is a linear combinations of the inputs, with the coefficients coming from the ith row of its matrix, rows of the matrix correspond to outputs of the $\texttt{FinVect}_k$ block. Because of this, the row operation subdiagrams in S' will have $0^\dagger$s immediately beneath them. Showing S' can be built from S reduces to showing composition of row operations with $0^\dagger$s builds the same number of $0^\dagger$s.

- Add a multiple c of one row to another row:

  If we want to add a multiple of the β row to the α row, we need a map $(y_\alpha, y_\beta) \mapsto (y_\alpha + cy_\beta, y_\beta)$. By the naturality of the braiding in a symmetric monoidal category, we can ignore any intermediate outputs:

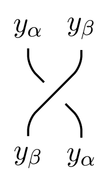


  When two cozeros are composed on the bottom of this diagram, the result is two cozeros:

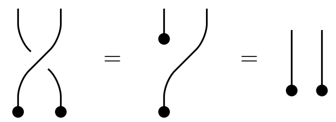


- Swap rows:

  If we want to swap the β row with the α row, we need a map $(y_\alpha, y_\beta) \mapsto (y_\beta, y_\alpha)$, which is the braiding of two outputs. Again, intermediate outputs may be ignored:

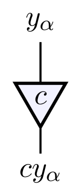


  When two cozeros are composed at the bottom of this diagram, the cut strings untwist by the naturality of the braiding:

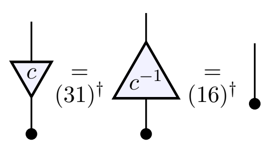


- Multiply a row by $c \neq 0$:

  The third row operation is multiplying an arbitrary row by a unit, but since k is a field, that means any $c \neq 0$. This is just the scaling map on one of the outputs:

  Because c is a unit, $c^{-1} \in k$, so s_c can be replaced by the adjoint of scaling by c^{-1}.

■

Given the PROP $\texttt{FinRel}_k$, it is natural to consider the free PROP $\texttt{SigFlow}_k$, which is defined by the same generators, but has no equations. The morphisms in this PROP are signal-flow diagrams[3]. The general considerations of Chapter 2 give a functor from $\texttt{SigFlow}_k$ to any other PROP that has the same generators, which 'imposes the equations' of the target PROP. In particular, we get a PROP morphism $\blacksquare\colon \texttt{SigFlow}_k \to \texttt{FinRel}_k$. In the state-space context, control theorists are only interested in a certain subcollection of signal-flow diagrams, which correspond to the state-space equations:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du,$$

[3]Because a free PROP must still respect the equations of the symmetry, our signal-flow diagrams are isomorphism classes of string diagrams.

where A, B, C and D are linear maps; and u, y, and x are input, output, and state vectors, respectively. We formalize this correspondence in Chapter 5. Intuitively, `SigFlow` is 'too big' and `FinRel` is 'too small', so we consider two ways to get a Goldilocks PROP: `Stateful` and `ContFlow`.

The PROP `ContFlow` is the most coarse subPROP of `SigFlow` whose morphisms include the subcollection of signal-flow diagrams that correspond to the state-space equations. We show `ContFlow` does not have any morphisms outside of this subcollection. While, roughly speaking, `ContFlow` is a way to 'shrink' `SigFlow`, `Stateful` is a way to 'grow' `FinRel`. These two approaches are explored in further detail in the following two chapters.

3.3 An example

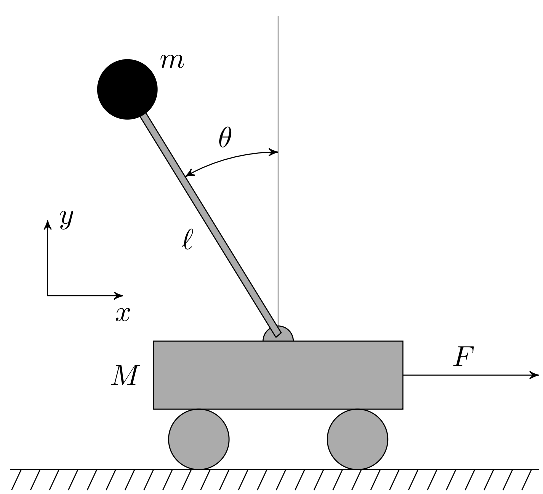


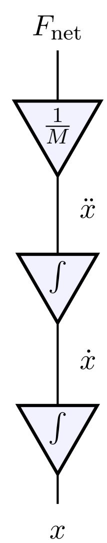
Figure 3.1: Schematic diagram of an inverted pendulum.

A famous example in control theory is the 'inverted pendulum': an upside-down pendulum on a cart [14]. The pendulum naturally tends to fall over, but we can stabilize it by setting up a feedback loop where we observe its position and move the cart back and forth in a suitable way based on this observation. Without introducing this feedback loop, let us see how signal-flow diagrams can be used to describe the pendulum and the cart. We shall see that the diagram for a system made of parts is built from the diagrams for the parts, not merely by composing and tensoring, but also with the help of duplication and coduplication, which give additional ways to set variables equal to one another.

Suppose the cart has mass M and can only move back and forth in one direction, so its position is described by a function $x(t)$. If it is acted on by a total force $F_{\mathrm{net}}(t)$ then Newton's second law says

$$F_{\mathrm{net}}(t) = M\ddot{x}(t).$$

We can thus write a signal-flow diagram with the force as input and the cart's position as output:

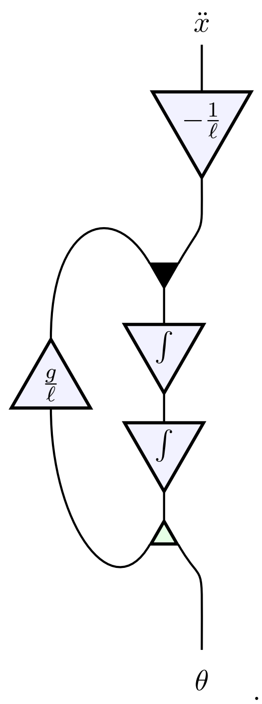


.

The inverted pendulum is a rod of length ℓ with a mass m at its end, mounted on the cart and only able to swing back and forth in one direction, parallel to the cart's movement. If its angle from vertical, $\theta(t)$, is small, then its equation of motion is approximately linear:

$$\ell\ddot{\theta}(t) = g\theta(t) - \ddot{x}(t),$$

where g is the gravitational constant. We can turn this equation into a signal-flow diagram with $\ddot{x}$ as input and θ as output:

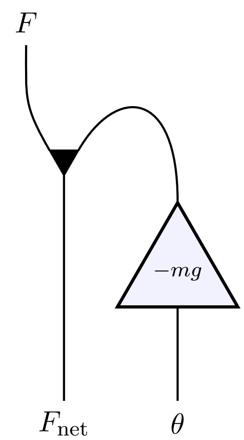


Note that this already includes a kind of feedback loop, since the pendulum's angle affects the force on the pendulum.

Finally, there is an equation describing the total force on the cart:

$$F_{\text{net}}(t) = F(t) - mg\theta(t),$$

where $F(t)$ is an externally applied force and $-mg\theta(t)$ is the force due to the pendulum. It will be useful to express this as follows:



.

Here we are treating θ as an output rather than an input, with the help of a cap.

The three signal-flow diagrams above describe the following linear relations:

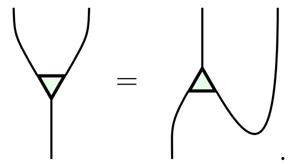
$$x = \int\int \frac{1}{M} F_{\mathrm{net}} \tag{3.1}$$

$$\theta = \int\int \left(\frac{g}{\ell}\theta - \frac{1}{\ell}\ddot{x} \right) \tag{3.2}$$

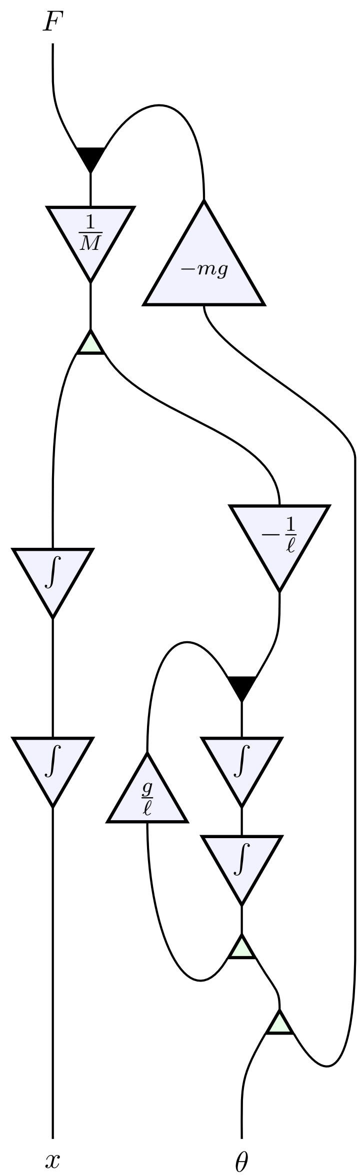
$$F_{\mathrm{net}} + mg\theta = F, \tag{3.3}$$

where we treat (3.1) as a linear relation with F_{net} as input and x as output, (3.2) as a linear relation with $\ddot{x}$ as input and θ as output, and (3.3) as a linear relation with F as input and $(F_{\mathrm{net}}, \theta)$ as output.

To understand how the external force affects the position of the cart and the angle of the pendulum, we wish to combine all three diagrams to form a signal-flow diagram that has the external force F as input and the pair (x, θ) as output. This is not just a simple matter of composing and tensoring the three diagrams. We can take F_{net}, which is an output of (3.3), and use it as an input for (3.1). But we also need to duplicate $\ddot{x}$, which appears as an intermediate variable in (3.1) since $\ddot{x} = \frac{1}{M} F_{\mathrm{net}}$, and use it as an input for (3.2). Finally, we need to take the variable θ, which appears as an output of both (3.2) and (3.3), and identify the two copies of this variable using coduplication. To emphasize the relational nature of the component, we shall write coduplication in terms of duplication and a cup, as follows:

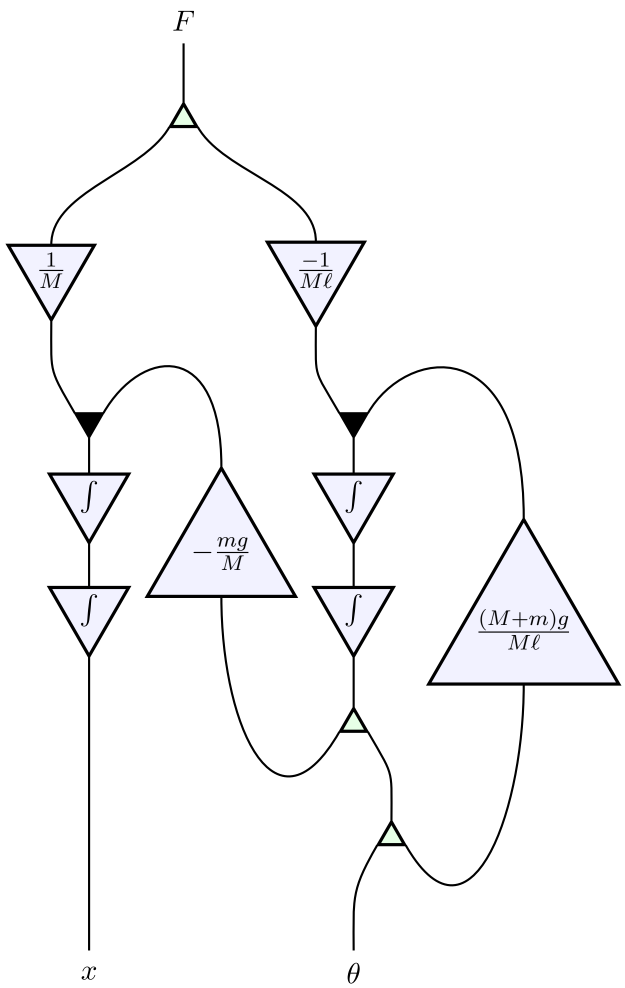


The result is this signal-flow diagram:



This is not the signal-flow diagram for the inverted pendulum that one sees in

Friedland's textbook on control theory [14]. We leave it as an exercise to the reader to rewrite the above diagram using the rules given in this paper, obtaining Friedland's diagram:



As a start, one can use Theorem 15 to prove that it is indeed possible to do this rewriting. To do this, simply check that both signal-flow diagrams define the same linear relation. The proof of the theorem gives a method to actually do the rewriting—but not necessarily the fastest method.

3.4 Related work

We conclude this chapter with some remarks aimed at setting it in context. This chapter is heavily based on [3], so we would like to focus on comparisons with other papers published around the same time. On April 30th, 2014, after much of [3] was written, Sobociński told Baez about some closely related papers that he wrote with Bonchi and Zanasi [6, 7]. These provide interesting characterizations of symmetric monoidal categories equivalent to $\mathrm{FinVect}_k$ and FinRel_k. Later, while [3] was being refereed, Wadsley and Woods [35] generalized the presentation of $\mathtt{FinVect}_k$ to the case where k is any commutative rig. We discuss Wadsley and Woods' work first, since doing so makes the exposition simpler.

What we have called $\mathtt{FinVect}_k$ here, Wadsley and Woods looked at from a slightly different perspective, getting an isomorphic PROP $\mathtt{Mat}(k)$, where a morphism $f\colon m \to n$ is an $n \times m$ matrix with entries in k, composition of morphisms is given by matrix multiplication, and the tensor product of morphisms is the direct sum of matrices. Wadsley and Woods gave an elegant description of the algebras of $\mathtt{Mat}(k)$. Suppose $\mathtt{P}$ is a PROP and $\mathtt{Q}$ is a strict symmetric monoidal category. Then the **category of algebras** of $\mathtt{P}$ in $\mathtt{Q}$ is the category of strict symmetric monoidal functors $F\colon \mathtt{P} \to \mathtt{Q}$ and natural transformations between these. If for every choice of $\mathtt{Q}$ the category of algebras of $\mathtt{P}$ in $\mathtt{Q}$ is equivalent to the category of algebraic structures of some kind in $\mathtt{Q}$, we say $\mathtt{P}$ is the PROP for structures of that kind.

In this language, Wadsley and Woods proved that $\mathtt{Mat}(k)$ is the PROP for 'bicommutative bimonoids over k'. To understand this, first note that for any bicommutative bimonoid A in $\mathtt{Q}$, the bimonoid endomorphisms of A can be added and composed, giving a rig $\mathrm{End}(A)$. A bicommutative bimonoid **over *k*** in $\mathtt{Q}$ is one equipped with a rig homomorphism $\Phi_A\colon k \to \mathrm{End}(A)$. Bicommutative bimonoids over k form a category where a morphism $f\colon A \to B$ is a bimonoid homomorphism compatible with this extra structure, meaning that for each $c \in k$ the square

$$\begin{array}{ccc} A & \xrightarrow{\Phi_A(c)} & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[\Phi_B(c)]{} & B \end{array}$$

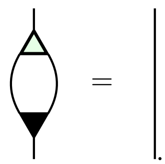
commutes. Wadsley and Woods proved that this category is equivalent to the category of algebras of $\mathtt{Mat}(k)$ in $\mathtt{Q}$.

This result amounts to a succinct restatement of Theorem 13, though technically the result is a bit different, and the style of proof much more so. The fact that an algebra of $\mathtt{Mat}(k)$ is a bicommutative bimonoid is equivalent to our equations **(1)–(10)**. The fact that $\Phi_A(c)$ is a bimonoid homomorphism for all $c \in k$ is equivalent to equations **(15)–(18)**, and the fact that Φ is a rig homomorphism is equivalent to equations **(11)–(14)**.

Even better, Wadsley and Woods showed that $\mathtt{Mat}(k)$ is the PROP for bicommutative bimonoids over k whenever k is a commutative rig. Subtraction and division are not required to define the PROP $\mathtt{Mat}(k)$, nor are they relevant to the definition of bicommutative bimonoids over k. Working with commutative rigs is not just generalization for the sake of generalization: it clarifies some interesting facts.

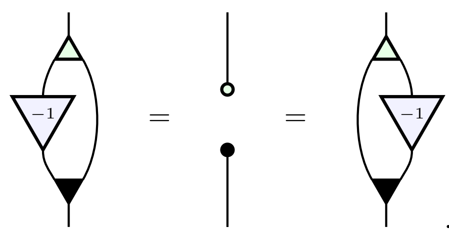
For example, the commutative rig of natural numbers gives a PROP $\mathtt{Mat}(\mathbb{N})$. This is equivalent to the symmetric monoidal category where morphisms are isomorphism classes of spans of finite sets, with disjoint union as the tensor product. Lack [22, Ex. 5.4] had already shown that this is the PROP for bicommutative bimonoids. But this also follows from the result of Wadsley and Woods, since every bicommutative bimonoid A is automatically equipped with a unique rig homomorphism $\Phi_A \colon \mathbb{N} \to \mathrm{End}(A)$.

Similarly, the commutative rig of booleans $\mathbb{B} = \{F, T\}$, with 'or' as addition and 'and' as multiplication, gives a PROP $\mathtt{Mat}(\mathbb{B})$. This is equivalent to the symmetric monoidal category where morphisms are relations between finite sets, with disjoint union as the tensor product. Mimram [26, Thm. 16] had already shown this is the PROP for **special** bicommutative bimonoids, meaning those where comultiplication followed by multiplication is the identity:



But again, this follows from the general result of Wadsley and Woods.

Finally, taking the commutative ring of integers $\mathbb{Z}$, Wadsley and Woods showed that $\mathtt{Mat}(\mathbb{Z})$ is the PROP for bicommutative Hopf monoids. The key here is that scaling by -1 obeys the axioms for an antipode, namely:



More generally, whenever k is a commutative ring, the presence of $-1 \in k$ guarantees that a bimonoid over k is automatically a Hopf monoid over k. So, when k is a commutative ring, Wadsley and Woods' result implies that $\mathtt{Mat}(k)$ is the PROP for Hopf monoids over k.

Earlier, Bonchi, Sobociński and Zanasi gave an elegant and very different proof that $\mathtt{Mat}(R)$ is the PROP for Hopf monoids over R when R is a principal ideal domain [6, Prop. 3.7]. The advantage of their argument is that they build up the PROP for Hopf monoids over R from smaller pieces, using some ideas developed by Lack [22].

These authors also proved that $\mathtt{FinRel}_k$ is a pushout in the category PROP of PROPs and PROP morphisms:

$$\begin{tikzcd}
\mathtt{Mat}(R) + \mathtt{Mat}(R)^{\mathrm{op}} \arrow[r] \arrow[d] & \mathrm{Span}(\mathtt{Mat}(R)) \arrow[d] \\
\mathrm{Cospan}(\mathtt{Mat}(R)) \arrow[r] & \mathtt{FinRel}_k \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

This pushout square requires a bit of explanation. Here R is any principal ideal domain whose field of fractions is k. For example, we could take $R = k$, though Bonchi, Sobociński and Zanasi are more interested in the example where $R = \mathbb{R}[s]$ and $k = \mathbb{R}(s)$. A morphism in $\mathrm{Span}(\mathtt{Mat}(R))$ is an isomorphism class of spans in $\mathtt{Mat}(R)$. There is a covariant functor

$$\begin{array}{ccc} \mathtt{Mat}(R) & \to & \mathrm{Span}(\mathtt{Mat}(R)) \\ m \xrightarrow{f} n & \mapsto & m \xleftarrow{1} m \xrightarrow{f} n \end{array}$$

and also a contravariant functor

$$\begin{array}{ccc} \mathtt{Mat}(R) & \to & \mathrm{Span}(\mathtt{Mat}(R)) \\ m \xrightarrow{f} n & \mapsto & n \xleftarrow{f} m \xrightarrow{1} m. \end{array}$$

Putting these together we get the functor from $\mathtt{Mat}(R) + \mathtt{Mat}(R)^{\mathrm{op}}$ to $\mathrm{Span}(\mathtt{Mat}(R))$ that gives the top edge of the square. Similarly, a morphism in $\mathrm{Cospan}(\mathtt{Mat}(R))$ is an isomorphism class of cospans in $\mathtt{Mat}(R)$, and we have both a covariant functor

$$\begin{array}{ccc} \mathtt{Mat}(R) & \to & \mathrm{Cospan}(\mathtt{Mat}(R)) \\ m \xrightarrow{f} n & \mapsto & m \xrightarrow{f} n \xleftarrow{1} n \end{array}$$

and a contravariant functor

$$\begin{array}{ccc} \mathtt{Mat}(R) & \to & \mathrm{Cospan}(\mathtt{Mat}(R)) \\ m \xrightarrow{f} n & \mapsto & n \xrightarrow{1} n \xleftarrow{f} m. \end{array}$$

Putting these together we get the functor from $\mathtt{Mat}(R) + \mathtt{Mat}(R)^{\mathrm{op}}$ to $\mathrm{Cospan}(\mathtt{Mat}(R))$ that gives the left edge of the square.

Bonchi, Sobociński and Zanasi analyze this pushout square in detail, giving explicit presentations for each of the PROPs involved, all based on their presentation of $\mathtt{Mat}(R)$. The upshot is a presentation of $\mathtt{FinRel}_k$ which is very similar to our presentation of $\mathtt{FinRel}_k$. Their methods allow them to avoid many, though not all, of the lengthy arguments that involve putting morphisms in standard form.

Chapter 4

The PROP $\mathtt{Stateful}$

4.1 Constructing categories of state

In the late 1950s and early 1960s, Kalman worked on the state-space approach to control theory: In 1960 [18] he introduced the concepts of *controllability* and *observability* into control theory, showing in 1963 [19] how these concepts can be used to decompose an arbitrary linear control system into four parts. He showed in the time-invariant case, these four parts are actual subsystems. Earlier work in control theory had focused on *transfer functions*, defined as the ratio of a transform (typically the Laplace transform) of the output of a system by the transform of the input of the system. Kalman showed transfer functions only capture the part of the system that is both controllable and observable: uncontrollable parts of the system do not depend on the input, and unobservable parts of the system do not affect the output.

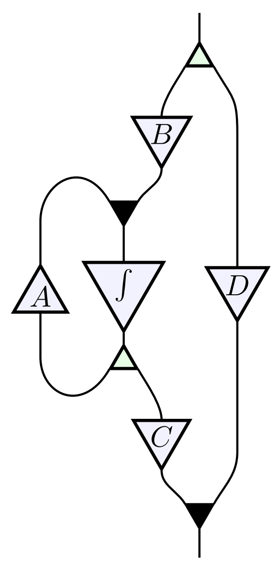
The morphisms of $\mathtt{FinVect}_k$ and $\mathtt{FinRel}_k$ are completely determined by how the input relates to the output, so while they are reasonable models for the frequency analysis approach and its transfer functions, they are unsatisfactory as models for the state-space approach. Our goal here is to define new categories based on $\mathtt{FinVect}_k$ and $\mathtt{FinRel}_k$ to address the shortcomings of these PROPs in the state-space context for linear time-invariant systems.

Kalman's concepts of controllability and observability apply to systems of differential equations of the form

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t),$$

where $u(t)$ is the input vector, $y(t)$ is the output vector, and $x(t)$ is the state vector. Written as a signal-flow diagram, such a system looks like



After taking Laplace transforms, we may also write the output y in terms of the input u simply as

$$y = (D + C(sI - A)^{-1}B)u.$$

Here the term Du gives the 'direct' dependence of output on input, while the other term, $C(sI - A)^{-1}B)u$, gives its 'indirect' dependence, mediated by the state x. We can visualize this split into direct and indirect terms as a noncommuting square

$$
\begin{tikzcd}
S \arrow[r, "(sI-A)^{-1}"] & T \arrow[d, "C"] \\
V_1 \arrow[u, "B"] \arrow[r, "D"'] & V_2
\end{tikzcd}
,
$$

where D goes directly from the vector space V_1 containing the input to the space V_2 containing the output, while the other arrows compose to give the 'indirect' map $C(sI - A)^{-1}B$. More abstractly, we can write such a square simply as

$$
\begin{tikzcd}
S \arrow[r, "a"] & T \arrow[d, "c"] \\
V_1 \arrow[u, "b"] \arrow[r, "d"'] & V_2
\end{tikzcd}
,
$$

Squares of this form serve as the morphisms in the PROPs we consider now.

4.2 The Box construction

In pursuit of this goal, we define a new construction that, for a suitable symmetric monoidal category $\mathcal{C}$, forms a new symmetric monoidal category $\Box\mathcal{C}$. The full details of how this works can be found in Appendix B. For this section we will focus on the particular case of the PROP $\Box(\mathtt{FinVect}_k)$.

Definition 16 *The category $\Box(\mathtt{FinVect}_k)$ has*

- *the same objects as $\mathtt{FinVect}_k$ (*i.e.* vector spaces k^n),*
- *morphisms that are equivalence classes of*

$$V_1 \xrightarrow{\Delta} V_1 \oplus V_1 \xrightarrow{\mathrm{id}_{V_1} \oplus b} V_1 \oplus S \xrightarrow{\mathrm{id}_{V_1} \oplus a} V_1 \oplus T \xrightarrow{d \oplus c} V_2 \oplus V_2 \xrightarrow{m} V_2 ,$$

abbreviated (d, c, a, b),

- *composition given by*

$$(d', c', a', b') \circ (d, c, a, b) = \left(d'd, [d'c \quad c'], \begin{bmatrix} a & 0 \\ a'b'ca & a' \end{bmatrix}, \begin{bmatrix} b \\ b'd \end{bmatrix} \right).$$

The morphisms of $\square(\texttt{FinVect}_k)$ can be depicted as non-commuting squares:

$$\begin{tikzcd} S \arrow[r, "a"] & T \arrow[d, "c"] \\ V_1 \arrow[u, "b"] \arrow[r, "d"'] & V_2 \end{tikzcd},$$

which explains the Box in the name of the category. Two squares, (d, c, a, b) and (d', c', a', b') are in the same equivalence class if there are isomorphisms $\alpha\colon S \to S'$ and $\omega\colon T \to T'$ in $\texttt{FinVect}_k$ such that the following diagram in $\texttt{FinVect}_k$ commutes:

$$\begin{tikzcd} S \arrow[rrr, "a"] \arrow[ddd, "\alpha"'] & & & T \arrow[ddl, "c"'] \arrow[ddd, "\omega"] \\ & & & \\ & V_1 \arrow[uul, "b"] \arrow[dl, "b'"] & V_2 & \\ S' \arrow[rrr, "a'"] & & & T' \arrow[ul, "c'"'] \end{tikzcd}.$$

Since $\square(\texttt{FinVect}_k)$ has the same objects as $\texttt{FinVect}_k$, it is clear that $\square(\texttt{FinVect}_k)$ is a skeletal category.

We refer to the objects S and T as the *prestate space* and *state space*, respectively. The formula for composition of morphisms in $\square(\texttt{FinVect}_k)$ can be understood as coming from the diagram:

$$
\begin{tikzcd}
 & & S_1 \oplus S_2 \arrow[rr, "a \oplus a'"] & & T_1 \oplus T_2 \arrow[dll, "\pi_1"'] \arrow[drr, "\pi_2"] & & \\
S_1 \arrow[urr, "\iota_1"] \arrow[rr, "a"] & & T_1 \arrow[dr, "c"'] & & S_2 \arrow[ull, "\iota_2"'] \arrow[rr, "a'"] & & T_2 \arrow[dl, "c'"] \\
 & V_1 \arrow[ul, "b"] \arrow[rr, "d"'] & & V_2 \arrow[ur, "b'"'] \arrow[rr, "d'"'] & & V_3 &
\end{tikzcd}
.
$$

The b and c sides in the composite come from the pentagons on the left and right, respectively. That is, b comes from summing the paths from V_1 to $S_1 \oplus S_2$, and c comes from summing the paths from $T_1 \oplus T_2$ to V_3. Similarly, the a side in the composite comes from the paths from $S_1 \oplus S_2$ to $T_1 \oplus T_2$, which includes the direct path $a \oplus a'$ (taking S_1 to T_1 and S_2 to T_2) and a looped path that goes through $a \oplus a'$ twice (taking S_1 to T_2). The d side in the composite comes from the most direct path from V_1 to V_3.

Theorem 17 *The category $\square(\texttt{FinVect}_k)$ is a monoidal category with direct sum of vector spaces as the monoidal product on objects of $\square(\texttt{FinVect}_k)$, and $(d, c, a, b) \oplus (d', c', a', b') = (d \oplus d', c \oplus c', a \oplus a', b \oplus b')$ as the monoidal product on morphisms.*

Proof. To show $\square(\texttt{FinVect}_k)$ is a category, we need to show the composition is well-defined and associative, and the unit laws hold. To show $\square(\texttt{FinVect}_k)$ is a monoidal category, we also need to show the associators and unitors exist and satisfy the pentagon and triangle equations. We start by showing composition in $\square(\texttt{FinVect}_k)$ is well-defined. Given two composable morphisms, $f \colon V_1 \to V_2$ and $g \colon V_2 \to V_3$ with representatives (d_1, c_1, a_1, b_1) and (d_1', c_1', a_1', b_1') for f and (d_2, c_2, a_2, b_2) and (d_2', c_2', a_2', b_2') for g, we have the following commutative diagrams:

$$
\begin{tikzcd}
S_1 \arrow[rrr, "a_1"] \arrow[dd, "\alpha_1"'] & & & T_1 \arrow[dl, "c_1"'] \arrow[dd, "\omega_1"] \\
& V_1 \arrow[ul, "b_1"] \arrow[dl, "b_1'"] & V_2 & \\
S_1' \arrow[rrr, "a_1'"] & & & T_1' \arrow[ul, "c_1'"']
\end{tikzcd}
$$

and

$$
\begin{tikzcd}
S_2 \arrow[rrr, "a_2"] \arrow[dd, "\alpha_2"'] & & & T_2 \arrow[dl, "c_2"'] \arrow[dd, "\omega_2"] \\
& V_2 \arrow[ul, "b_2"] \arrow[dl, "b_2'"] & V_3 & \\
S_2' \arrow[rrr, "a_2'"] & & & T_2' \arrow[ul, "c_2'"']
\end{tikzcd}
.
$$

We leave it as an exercise to the reader to show $\alpha_{12} = \alpha_1 \oplus \alpha_2 \colon S_1 \oplus S_2 \to S_1' \oplus S_2'$ and $\omega_{12} = \omega_1 \oplus \omega_2 \colon T_1 \oplus T_2 \to T_1' \oplus T_2'$ are the isomorphisms required to make the corresponding diagram for $g \circ f$ commute.

It is easy to verify that

$$
\begin{tikzcd}
0 \arrow[r] & 0 \arrow[d] \\
V \arrow[u] \arrow[r, "1_V"'] & V
\end{tikzcd}
$$

is a left and right identity morphism in $\square(\texttt{FinVect}_k)$. The associators and unitors can be formed by the same trick, but these (along with the pentagon and triangle equations) are trivial since $\texttt{FinVect}_k$ is a strict monoidal category. It is also easy to see the monoidal product on morphisms is compatible with composition, so it remains to show composition in $\square(\texttt{FinVect}_k)$ is associative.

Aside from the source and target objects, morphisms in $\square(\texttt{FinVect}_k)$ have six pieces of data: the prestate space, the state space, and four linear maps, d, c, a, and b.

To check associativity in $\square(\texttt{FinVect}_k)$, we need to ensure both groupings of $f_3 \circ f_2 \circ f_1$ of composable morphisms in $\square(\texttt{FinVect}_k)$ give the same results for all six of these pieces of data. The prestate space and the state space will be the same, thanks to the associativity of the monoidal product in $\texttt{FinVect}_k$. Denoting the compositions $(d_j, c_j, a_j, b_j) \circ (d_i, c_i, a_i, b_i)$ as $(d_{ij}, c_{ij}, a_{ij}, b_{ij})$, we get $(d_3, c_3, a_3, b_3) \circ (d_{12}, c_{12}, a_{12}, b_{12}) = (d_{12,3}, c_{12,3}, a_{12,3}, b_{12,3})$:

$$\begin{tikzcd}
S_1 \oplus S_2 \arrow[r, "a_{12}"] & T_1 \oplus T_2 \arrow[dr, "c_{12}"] & & S_3 \arrow[r, "a_3"] & T_3 \arrow[dl, "c_3"] \\
V_1 \arrow[u, "b_{12}"] \arrow[rr, "d_{12}"'] & & V_3 \arrow[ur, "b_3"] \arrow[r, "d_3"'] & V_4 &
\end{tikzcd}
=
\begin{tikzcd}
(S_1 \oplus S_2) \oplus S_3 \arrow[r, "a_{12,3}"] & (T_1 \oplus T_2) \oplus T_3 \arrow[d, "c_{12,3}"] \\
V_1 \arrow[u, "b_{12,3}"] \arrow[r, "d_{12,3}"'] & V_4
\end{tikzcd}$$

and $(d_{23}, c_{23}, a_{23}, b_{23}) \circ (d_1, c_1, a_1, b_1) = (d_{1,23}, c_{1,23}, a_{1,23}, b_{1,23})$:

$$\begin{tikzcd}
S_1 \arrow[r, "a_1"] & T_1 \arrow[dr, "c_1"] & & S_2 \oplus S_3 \arrow[r, "a_{23}"] & T_2 \oplus T_3 \arrow[dl, "c_{23}"] \\
X_1 \arrow[u, "b_1"] \arrow[rr, "d_1"'] & & X_2 \arrow[ur, "b_{23}"] \arrow[r, "d_{23}"'] & X_4 &
\end{tikzcd}
=
\begin{tikzcd}
S_1 \oplus (S_2 \oplus S_3) \arrow[r, "a_{1,23}"] & T_1 \oplus (T_2 \oplus T_3) \arrow[d, "c_{1,23}"] \\
X_1 \arrow[u, "b_{1,23}"] \arrow[r, "d_{1,23}"'] & X_4
\end{tikzcd}.$$

For the linear map data, d, c, a, and b, the associativity of $\square(\texttt{FinVect}_k)$ requires $d_{12,3} = d_{1,23}$, $c_{12,3} = c_{1,23}$, $a_{12,3} = a_{1,23}$, and $b_{12,3} = b_{1,23}$. It is clear that the associativity requirement for d is met because composition of linear maps is associative in $\texttt{FinVect}_k$ — $d_{1,23} = d_1(d_2d_3) = (d_1d_2)d_3 = d_{12,3}$. We see the associativity requirement for a is met because $a_{ij} = \begin{bmatrix} a_i & 0 \\ a_jb_jc_ia_i & a_j \end{bmatrix}$, which means

$$a_{12,3} = \begin{bmatrix} a_{12} & 0 \\ a_3b_3c_{12}a_{12} & a_3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_1 & 0 \\ a_2b_2c_1a_1 & a_2 \end{bmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ [a_3b_3d_2c_1a_1 + a_3b_3c_2a_2b_2c_1a_1 \quad a_3b_3c_2a_2] & a_3 \end{bmatrix},$$

since $c_{12} = [d_2c_1 \quad c_2]$. A similar calculation gives the same matrix, grouped slightly differently, for $a_{1,23}$,

$$a_{1,23} = \begin{bmatrix} a_1 & 0 & 0 \\ \begin{bmatrix} a_2b_2c_1a_1 \\ a_3b_3d_2c_1a_1 + a_3b_3c_2a_2b_2c_1a_1 \end{bmatrix} & \begin{bmatrix} a_2 & 0 \\ a_3b_3c_2a_2 & a_3 \end{bmatrix} \end{bmatrix}.$$

The proofs that c and b meet their respective associativity requirements are similar to each other, transposed. We present the argument for c and leave the argument for b to the reader. Since $c_{ij} = [d_jc_i \quad c_j]$, we have

$$\begin{aligned} c_{12,3} &= [d_3c_{12} \quad c_3] \\ &= [d_3[d_2c_1 \quad c_2] \quad c_3] \\ &= [[d_3d_2c_1 \quad d_3c_2] \quad c_3] \\ &= [d_3d_2c_1 \quad [d_3c_2 \quad c_3]] \\ &= [d_{23}c_1 \quad c_{23}] = c_{1,23}. \end{aligned}$$

So we see composition of morphisms in $\square(\texttt{FinVect}_k)$ is associative. ■

These kinds of manipulations of $\square(\texttt{FinVect}_k)$ make more sense when $\square(\texttt{FinVect}_k)$ is understood as a category with an obvious evaluation functor eval: $\square(\texttt{FinVect}_k) \to \texttt{FinVect}_k$. We can also find a 'feedthrough' functor feed: $\square(\texttt{FinVect}_k) \to \texttt{FinVect}_k$ and a functor in the reverse direction Box: $\texttt{FinVect}_k \to \square(\texttt{FinVect}_k)$. The map of objects $\mathrm{eval}_0\colon \mathrm{Ob}(\square(\texttt{FinVect}_k)) \to \mathrm{Ob}(\texttt{FinVect}_k)$ is trivial, $\mathrm{feed}_0 = \mathrm{eval}_0$, and Box_0 is its inverse. The map of morphisms $\mathrm{eval}_1\colon \mathrm{Mor}(\square\mathcal{C}) \to \mathrm{Mor}(\mathcal{C})$ is given by $\mathrm{eval}_1(d,c,a,b) = d + cab$, $\mathrm{feed}_1\colon \mathrm{Mor}(\square\mathcal{C}) \to \mathrm{Mor}(\mathcal{C})$ is given by $\mathrm{feed}_1(d,c,a,b) = d$, and $\mathrm{Box}_1\colon \mathrm{Mor}(\mathcal{C}) \to \mathrm{Mor}(\square\mathcal{C})$ is given by $\mathrm{Box}_1(d) = (d, !, 0, 0)$. That is,

$$\mathrm{eval}_1\left(\begin{tikzcd} S \arrow[r, "a"] & T \arrow[d, "c"] \\ V_1 \arrow[u, "b"] \arrow[r, "d"'] & V_2 \end{tikzcd}\right) = d + cab, \quad \mathrm{feed}_1\left(\begin{tikzcd} S \arrow[r, "a"] & T \arrow[d, "c"] \\ V_1 \arrow[u, "b"] \arrow[r, "d"'] & V_2 \end{tikzcd}\right) = d$$

$$\text{and } \mathrm{Box}_1(d) = \begin{tikzcd} 0 \arrow[r] & 0 \arrow[d] \\ V_1 \arrow[u] \arrow[r, "d"'] & V_2 \end{tikzcd}.$$

Theorem 18 *There are functors* eval, feed, *and* Box *as defined above which are* PROP *morphisms, and*

- eval *and* feed *are full, but not faithful,*
- Box *is faithful, but not full,*
- Box *does not preserve limits or colimits. In particular,* Box *has no adjoint.*

Proof. It is easy to check that eval, feed, and Box all preserve identity maps. Preservation of composition is again easy to check for feed and Box, but eval takes a little more work. On one hand, $\mathrm{eval}(d, c, a, b) \circ \mathrm{eval}(d', c', a', b') = (d + cab) \circ (d' + c'a'b') = dd' + dc'a'b' + cabd' + cabc'a'b'$. On the other hand,

$$\begin{aligned} \mathrm{eval}((d,c,a,b) \circ (d',c',a',b')) &= \mathrm{eval}\left(\left(dd', [dc' \quad c], \begin{bmatrix} a' & 0 \\ abc'a' & a \end{bmatrix}, \begin{bmatrix} b' \\ bd' \end{bmatrix}\right)\right) \\ &= dd' + [dc' \quad c] \begin{bmatrix} a' & 0 \\ abc'a' & a \end{bmatrix} \begin{bmatrix} b' \\ bd' \end{bmatrix} \\ &= dd' + dc'a'b' + cabc'a'b' + cabd'. \end{aligned}$$

Addition is commutative, so eval preserves composition.

All three functors act as identities on objects, so it immediately follows they are essentially surjective. We note that feed$\circ$Box and eval$\circ$Box are both the identity functor on $\mathtt{FinVect}_k$, which implies feed and eval are surjective on *all* morphisms, hence full. This also implies Box is injective on morphisms, so Box is faithful. On the other hand, a morphism in $\square(\mathtt{FinVect}_k)$ between $\mathrm{Box}_0(V_1)$ and $\mathrm{Box}_0(V_2)$ where the prestate space or state space are not isomorphic to the zero object is not the Box_1-image of any morphism in $\mathtt{FinVect}_k$, so Box is not full. Similarly, feed and eval cannot be faithful.

The object 0 in $\mathtt{FinVect}_k$ is both initial and terminal, so to show Box does not preserve limits or colimits, it suffices to show $\mathrm{Box}_0(0)$ is neither initial nor terminal. For each $f\colon S \to T$ in $\mathtt{FinVect}_k$, there will be a morphism $0_f\colon \mathrm{Box}_0(0) \to \mathrm{Box}_0(0)$ in $\Box(\mathtt{FinVect}_k)$, given by

$$0_f = \begin{tikzcd} S \arrow[r, "f"] & T \arrow[d] \\ 0 \arrow[u] \arrow[r] & 0 \end{tikzcd}.$$

Thus Box does not preserve initial or terminal objects, so it cannot preserve limits or colimits. Because Box does not preserve limits, Box is not a right adjoint, and because Box does not preserve colimits, Box is not a left adjoint.

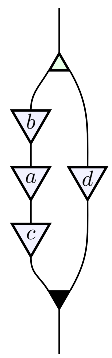
$$\begin{tikzcd}[column sep=small] & V_2 \oplus V_3 \oplus V_1 & & & & V_2 \oplus V_3 \oplus V_1 & \\ V_1 \oplus V_2 \oplus V_3 \arrow[ur, "B_{V_1,V_2\oplus V_3}"] \arrow[rr, "B_{V_1,V_2}\oplus \mathrm{Id}"'] & & V_2 \oplus V_1 \oplus V_3 \arrow[ul, "\mathrm{Id}\oplus B_{V_1,V_3}"'] & \arrow[r, shift left, "\mathrm{Box}"] & \arrow[l, shift left, "\mathrm{eval}"] V_1 \oplus V_2 \oplus V_3 \arrow[ur, "(B_{V_1,V_2\oplus V_3},!,0,0)"] \arrow[rr, "(B_{V_1,V_2}\oplus \mathrm{Id},!,0,0)"'] & & V_2 \oplus V_1 \oplus V_3 \arrow[ul, "(\mathrm{Id}\oplus B_{V_1,V_3},!,0,0)"'] \end{tikzcd}$$

Figure 4.1: A hexagon law is preserved by eval and Box. Since $\mathtt{FinVect}_k$ and $\Box(\mathtt{FinVect}_k)$ are strict monoidal categories, the associators are all identities, so three sides of the hexagon have been omitted.

It remains to show these functors are *symmetric* monoidal functors. We already know $\mathtt{FinVect}_k$ and $\Box(\mathtt{FinVect}_k)$ are strict monoidal categories, so the associators and unitors are all identity morphisms. It is easy to check the symmetry isomorphisms in $\Box(\mathtt{FinVect}_k)$ are the Box_1-images of the symmetry isomorphisms in $\mathtt{FinVect}_k$. Since $(d,!,0,0)\circ(d',!,0,0) = (d\circ d',!,0,0)$ in $\Box(\mathtt{FinVect}_k)$, any coherence law in $\mathtt{FinVect}_k$ is preserved by Box. Similarly, $\mathrm{feed}_1(d,!,0,0) = \mathrm{eval}_1(d,!,0,0) = d$, so eval and feed will also preserve the coherence laws in $\Box(\mathtt{FinVect}_k)$ that come from $\mathtt{FinVect}_k$. See Figure 4.1 for an example of a coherence law preserved by these three functors.

Thus we see all three functors are identity on objects symmetric monoidal functors between PROPs, hence PROP morphisms. ■

An alternate way to depict morphisms in $\square(\texttt{FinVect}_k)$ is through string diagrams. The morphism (d, c, a, b) is depicted:



.

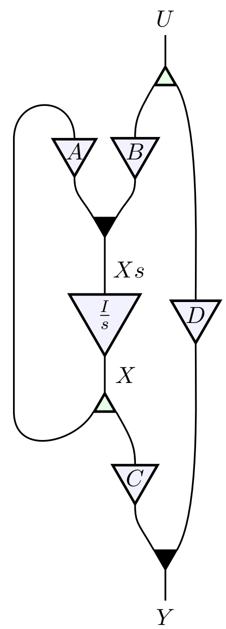
This form will make it easier to see the connection between the state equations and the PROP $\texttt{Stateful}_k$, which we introduce in the next section.

4.3 The PROP $\texttt{Stateful}$

Recall the state equations, Equation 1.1 and Equation 1.2, and the associated convention that $\dim(u) = m$, $\dim(x) = n$, and $\dim(y) = p$. Note that the Laplace transform of these equations give the following:

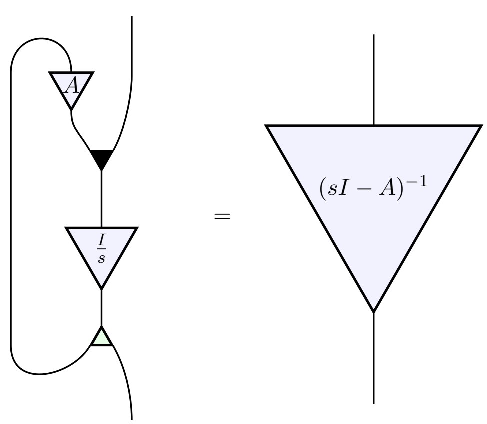
$$Xs - x(0) = AX + BU$$
$$Y = CX + DU,$$

where U, X, and Y are the Laplace transforms of u, x, and y, respectively. Under the assumption that the initial state vector is zero, these equations can be expressed as a morphism in $\texttt{FinRel}_{k(s)}$ with the following signal-flow diagram:



,

where the subdiagrams A, B, C, and D are the standard forms for their respective linear maps in the state equations, and I is the identity on k^n. Note that each instance of addition (*resp.* duplication) in the diagram represents zero or more copies of the morphism, in parallel. This diagram can always be rewritten in a form without cups or caps using the reduction:



.

Since A has no dependence on s in the time-independent case, $sI - A$ will have an inverse in $\texttt{FinVect}_{k(s)}$[1]. Similarly, given $a = (sI - A)^{-1}$, we can find $A = sI - a^{-1}$, so a and A

[1]More generally, as long as the Laplace transform of A has no positive powers of s, $sI - A$ can be guaranteed invertible. Even in the case where A depends on time, its Laplace transform will only ever

both provide the same information.

Note that we have the right form for a morphism in $\square(\texttt{FinVect}_{k(s)})$, where $a = (sI - A)^{-1}, b = B, c = C, d = D$, but not all morphisms in $\square(\texttt{FinVect}_{k(s)})$ arise this way, *e.g.* $(s, !, 0, 0)$. This motivates the definition of $\texttt{Stateful}_k$ as the subPROP of $\square(\texttt{FinVect}_{k(s)}))$ where the morphisms are of this form.

Definition 19 *The category* $\texttt{Stateful}_k$ *is the subPROP of* $\square(\texttt{FinVect}_{k(s)})$ *with*

- *the same objects as* $\texttt{FinVect}_{k(s)}$
- *morphisms of the form* $(D, C, (sI - A)^{-1}, B)$ *for some* $A, B, C, D \in \texttt{FinVect}_k$.

Proposition 20 *The category* $\texttt{Stateful}_k$ *is actually a PROP.*

Proof. For this definition of $\texttt{Stateful}_k$ to make sense, composition in $\texttt{Stateful}_k$ must be closed: $(D, C, (sI - A)^{-1}, B) \circ (D', C', (sI' - A')^{-1}, B') = (D'', C'', (sI'' - A'')^{-1}, B'')$ for some $A'', B'', C'', D'' \in \texttt{FinVect}_k$. We leave it to the reader to verify

$$A'' = \begin{bmatrix} A & 0 \\ B'C & A' \end{bmatrix},$$

where $I'' = I \oplus I'$. Closure for the other three sides is immediate. ■

In Appendix B the Box construction is extended to allow, among other possibilities, $\mathrm{FinVect}_k$ to replace $\texttt{FinVect}_k$. While $\square(\mathrm{FinVect}_k)$ is no longer strict, it is still symmetric monoidal. Thus we can define $\mathrm{Stateful}_k$ as a symmetric monoidal subcategory of $\square(\mathrm{FinVect}_{k(s)})$ similarly to how $\texttt{Stateful}_k$ is defined here. Indeed, $\texttt{Stateful}_k$ is the skeletalization of $\mathrm{Stateful}_k$.

contain positive powers of s if a distribution like $\frac{\mathrm{d}}{\mathrm{d}t}\delta(t)$ appears in A.

4.4 Controllability and observability

Controllability and observability were introduced based on a physical interpretation of the solutions to Equations 1.1 and 1.2. The results in this section have long been well-known in the control theory literature [18, 19]. When the matrices $A(t)$ and $B(t)$ are continuous in t, the general solution to Equation 1.1 has the form

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)u(\tau)d\tau,$$

where $\Phi(t, \tau)$ is a fundamental matrix of solutions satisfying $\Phi(t, t) = I$ for all t. A system is **controllable** if for each state x and time t_0 there is an input function $u(t)$ such that the state can be set to the equilibrium state in a finite amount of time. That is, there is a function $u(t)$ such that $x(t_1) = 0$ for some $t_1 > t_0$. An equivalent characterization of controllability involves a symmetric matrix called the *Controllability Gramian*, $W_c(t_0, t_1)$. A system is controllable if and only if

$$W_c(t_0, t_1) = \int_{t_0}^{t_1} \Phi(t_0, t)B(t)B^\top(t)\Phi^\top(t_0, t)dt$$

is positive definite for some $t_1 > t_0$. In the case of linear time-invariant systems, this criterion is simplified to finding the row rank of the block matrix $M_c = [B, AB, \ldots, A^{n-1}B]$. This controllability matrix M_c is an $n \times mn$ matrix, and a linear time-invariant system is controllable when

$$\text{rank}(M_c) = n.$$

While controllability of a system ignores the system's output, observability of a system ignores the system's input, which can be accomplished by setting the input $u(t) = 0$. A system is **observable** if the state $x(t_0)$ of the system can be determined at some later time t_1 by setting the input function $u(t)$ to zero and measuring the output function $y(t)$. Kalman noticed a 'duality principle' connecting controllability and observability. If you

1. Reverse the direction of time,

2. Swap input and output constraints,

3. Replace Φ with $\Phi^\top$,

then an observable system is transformed into a controllable system and vice versa. Explicitly, this duality transforms Equations 1.1 and 1.2 by making the following replacements:

$$\begin{gathered}t - t_0 = t_0 - t', \\ A(t - t_0) \Leftrightarrow A^\top(t_0 - t'), \\ B(t - t_0) \Leftrightarrow C^\top(t_0 - t'), \\ C(t - t_0) \Leftrightarrow B^\top(t_0 - t'), \\ D(t - t_0) \Leftrightarrow D^\top(t_0 - t').\end{gathered}$$

From this duality, we can see the *Observability Gramian* will be

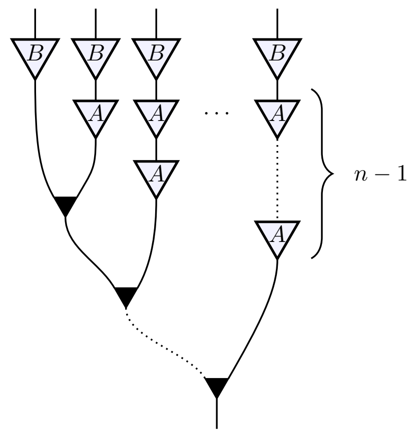
$$W_o(t_0, t_1) = \int_{t_0}^{t_1} \Phi^\top(t, t_1) C^\top(t) C(t) \Phi(t, t_1) dt,$$

which says a system is observable when $W_o(t_0, t_1)$ is positive definite for some $t_1 > t_0$. We also see the observability matrix for a linear time-invariant system will be $M_o = [C, CA, \ldots, CA^{n-1}]^\top$. This observability matrix M_o is an $np \times n$ matrix, and a linear time-invariant system is determined to be observable using the column rank:

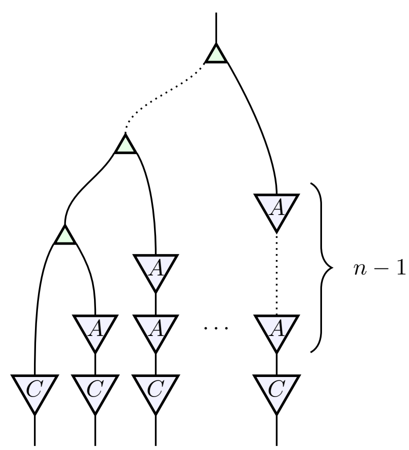
$$\text{rank}(M_o) = n.$$

If we view M_c as a linear transformation $\mathbb{R}^{mn} \to \mathbb{R}^n$, a controllable system has $\text{rank}(M_c) = n$, which means M_c is an epimorphism for a controllable system. Similarly, when $M_o \colon \mathbb{R}^n \to \mathbb{R}^{np}$ is viewed as a linear transformation, the system is observable exactly when M_o is a monomorphism.

Diagrammatically, this can be expressed as saying that a stateful morphism $(D, C, (sI - A)^{-1}, B)$ is controllable if:

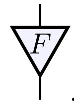


is an epimorphism in $\mathtt{FinVect}_k$, and it is observable if:



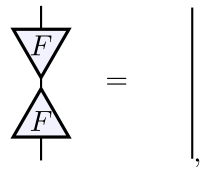
is a monomorphism in $\mathtt{FinVect}_k$.

Sobociński noted [32] there are purely diagrammatic tests for determining whether a linear relation is an epimorphism and whether it is a monomorphism. Given a linear map $F\colon V \to W$, it is a monomorphism if $F^\dagger F = 1_V$ and an epimorphism if $FF^\dagger = 1_W$. This extends to linear relations. Diagrammatically, given a linear relation F, depicted for convenience as

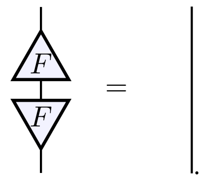


,

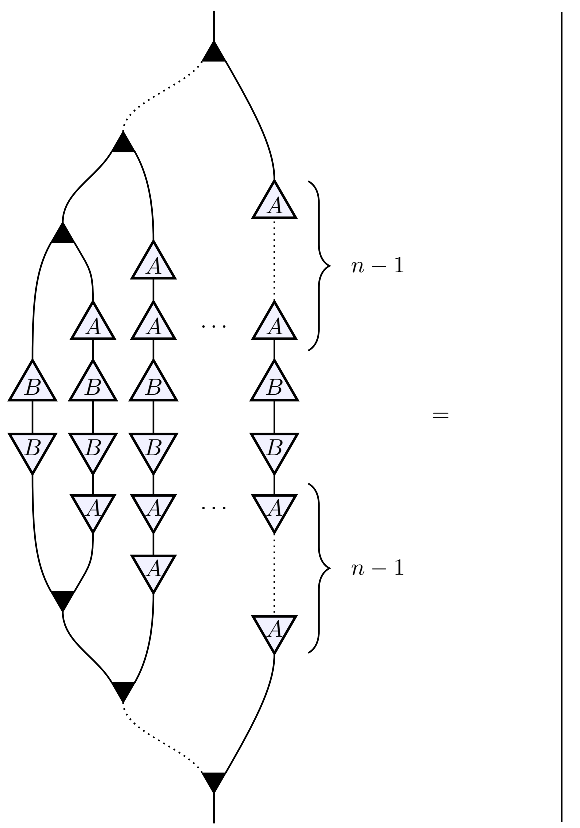
F is a monomorphism if



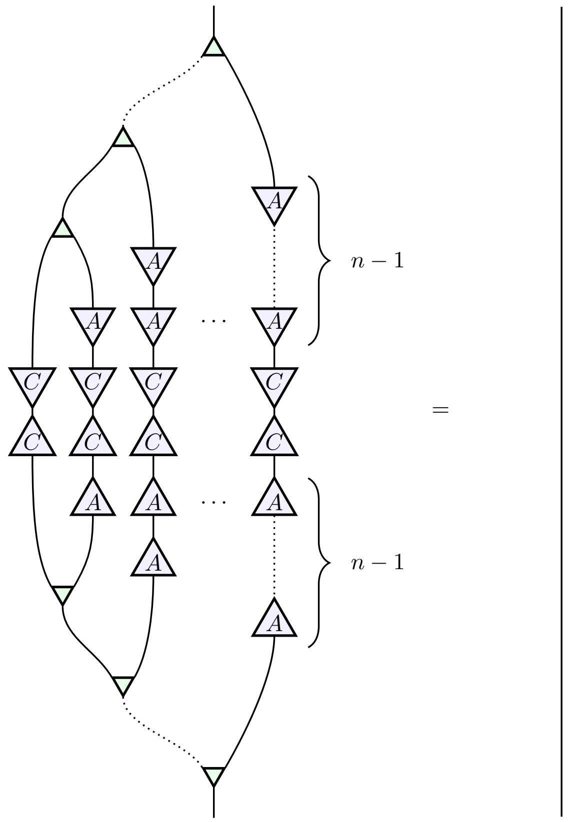
and F is an epimorphism if



Combining these results, we can say a stateful morphism $(D, C, (sI - A)^{-1}, B)$ is controllable when



and observable when



While the controllability and observability criteria deal with linear maps (morphisms in $\texttt{FinVect}_k$), the detour through $\texttt{Stateful}_k$ is still necessary. The linear maps A, B, and C are defined in terms of stateful morphisms: a linear map in $\texttt{FinVect}_{k(s)}$ has no way of 'knowing' what A, B, and C are. At best, a state space of minimum dimension can be determined. Through the rose-tinted glasses of $\texttt{FinVect}_{k(s)}$ alone, every morphism appears to be both controllable and observable!

Chapter 5

The PROP ContFlow

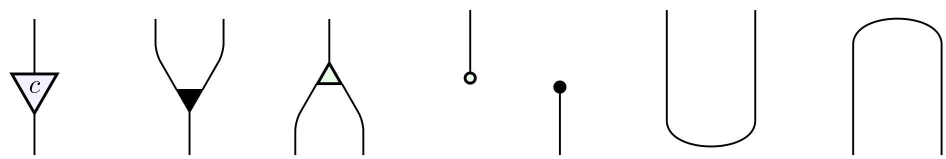
In this chapter we will define the PROP $\texttt{ContFlow}_k$ of 'good' signal-flow diagrams over a field k. Roughly speaking, a 'good' signal-flow diagram is one for which we can describe controllability and observability via stateful morphisms, using the results from Section 4.4. A sufficient condition on signal-flow diagrams for controllability and observability to make sense, then, is for the signal-flow diagram to be a morphism in $\texttt{ContFlow}_k$. A more formal description of what constitutes a 'good' signal-flow diagram involves the commutative square in Figure 5.1. The limitations of $\texttt{Stateful}_k$ force limitations on the

$$\begin{tikzcd} \texttt{ContFlow}_k \arrow[r, "\lozenge"] \arrow[d, hook, "j"'] & \texttt{Stateful}_k \arrow[d, "i \circ \mathrm{eval}"] \\ \texttt{SigFlow}_{k,s} \arrow[r, "\blacksquare"'] & \texttt{FinRel}_{k(s)} \end{tikzcd}$$

Figure 5.1: This square in PROP commutes.

kinds of signal-flow diagrams that can be considered 'good' here. The simplest signal-flow diagrams that fail to be 'good' are the generators $\cup$ and $\cap$. Indeed, control theorists never write these explicitly, and only implicitly use them as parts of larger signal-flow diagrams.

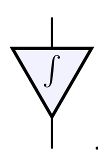
In order to formalize this discussion, we consider $\mathtt{SigFlow}_k$, the free PROP of all signal-flow diagrams over a field k, which we informally referred to at the end of Section 3.2. Recall, the signal-flow diagrams that generate this PROP are denoted



,

where $c \in k$ is arbitrary. There is an obvious 'black-box' functor $\blacksquare \colon \mathtt{SigFlow}_k \to \mathtt{FinRel}_k$ which is also a PROP morphism, thanks to the machinery of Chapter 2.

More generally, suppose S is any subset of k. Then we can define $\mathtt{SigFlow}_S$ to be the free PROP on the above generators, where c is restricted to be an element of S. This is a subPROP of $\mathtt{SigFlow}_k$, so we can restrict the black-box functor $\blacksquare \colon \mathtt{SigFlow}_k \to \mathtt{FinRel}_k$ and obtain a PROP morphism, which by abuse of notation we call $\blacksquare \colon \mathtt{SigFlow}_S \to \mathtt{FinRel}_k$ and let the context indicate which $\blacksquare$ is intended. Note that when S generates k as a field, the black-box functor from $\mathtt{SigFlow}_S$ is full. This abuse will be used primarily for $k \cup \{\frac{1}{s}\}$ as a subset of $k(s)$, with $\mathtt{SigFlow}_{k\cup\{\frac{1}{s}\}}$ abbreviated as $\mathtt{SigFlow}_{k,s}$. We call $\mathtt{SigFlow}_{k,s}$ the PROP of signal-flow diagrams with integrators. Integrators are treated separately from the other scalings here because integration plays a special role in control theory. Compared to $\mathtt{SigFlow}_k$, $\mathtt{SigFlow}_{k,s}$ is a free PROP that has one extra generator to indicate integration:



.

In Theorem 18 we found some PROP morphisms from $\mathtt{Stateful}_k$ to $\mathtt{FinVect}_{k(s)}$, namely eval and feed. By composing eval with the inclusion map $i \colon \mathtt{FinVect}_{k(s)} \to \mathtt{FinRel}_{k(s)}$, we get a PROP morphism $i \circ \mathrm{eval} \colon \mathtt{Stateful}_k \to \mathtt{FinRel}_{k(s)}$ that will be instrumental in determining which signal-flow diagrams are 'good'. Ultimately we will find a PROP morphism $\lozenge\colon \texttt{ContFlow}_k \to \texttt{Stateful}_k$ and a commutative square in the category PROP shown in Figure 5.1. The desire to find the PROP $\texttt{ContFlow}_k$ that makes this square commute leads us directly to the definition of $\texttt{ContFlow}_k$, stated in Definition 31.

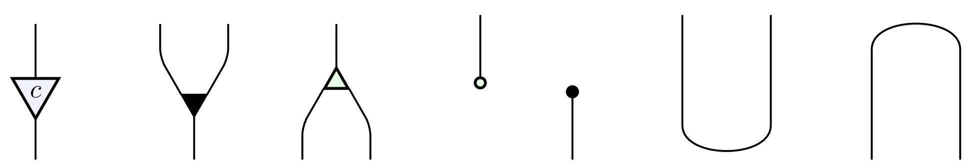
5.1 Finding $\texttt{ContFlow}_k$

Since we wish to define $\texttt{ContFlow}_k$ as a PROP of signal-flow diagrams with a certain property, we first define PROPs of signal-flow diagrams and then narrow down to those with that property.

Definition 21 *The PROP* $\texttt{SigFlow}_{\boldsymbol{k}}$ *is the free PROP on the generators of* $\texttt{FinRel}_k$*. That is,* $\texttt{SigFlow}_k = \texttt{F}(\Sigma_{\texttt{FinRel}_k})$*, where*

$$\begin{aligned}\Sigma_{\texttt{FinRel}_k} = \{&\sigma_+\colon 2 \to 1, \sigma_0\colon 0 \to 1, \sigma_\Delta\colon 1 \to 2, \sigma_!\colon 1 \to 0, \sigma_\cup\colon 2 \to 0, \sigma_\cap\colon 0 \to 2\} \\ &\cup \{\sigma_{s_c}\colon 1 \to 1 : c \in k\},\end{aligned}$$

so that the $\texttt{F}$*-images of these formal symbols are the generators listed in Lemma 14:*



where $c \in k$ is arbitrary.

For any subset S of the field k, we also have the following definition.

Definition 22 *The PROP* $\texttt{SigFlow}_{\boldsymbol{S}}$ *is the free PROP on the generators of* $\texttt{FinRel}_k$*, with the scaling morphisms s_c restricted to $c \in S$. That is,* $\texttt{SigFlow}_S = \texttt{F}(\Sigma_{\texttt{FinRel}_S})$*, where*

$$\begin{aligned}\Sigma_{\texttt{FinRel}_S} = \{&\sigma_+\colon 2 \to 1, \sigma_0\colon 0 \to 1, \sigma_\Delta\colon 1 \to 2, \sigma_!\colon 1 \to 0, \sigma_\cup\colon 2 \to 0, \sigma_\cap\colon 0 \to 2\} \\ &\cup \{\sigma_{s_c}\colon 1 \to 1 : c \in S\},\end{aligned}$$

so that the $\mathtt{F}$*-images of these formal symbols are the generators listed in Definition 21 above, but now with* $c \in S$*.*

Since $\mathtt{SigFlow}_k = \mathtt{F}\Sigma_{\mathtt{FinRel}_k}$, there is an obvious PROP morphism $\blacksquare\colon \mathtt{SigFlow}_k \to \mathtt{FinRel}_k$, namely the coequalizer of $\mathtt{F}\mathcal{E}_{\mathtt{FinRel}_k} \rightrightarrows \mathtt{F}\Sigma_{\mathtt{FinRel}_k}$. Following Lawvere's ideas on functorial semantics [24], we can treat the PROP $\mathtt{SigFlow}_k$ as providing 'syntax' and the PROP $\mathtt{FinRel}_k$ as providing 'semantics'. In other words, morphisms in $\mathtt{SigFlow}_k$ are a notation—signal-flow diagrams—while morphisms in $\mathtt{FinRel}_k$ are what this notation stands for, namely linear relations between inputs and outputs, which we arrive at by imposing the equations of $\mathtt{FinRel}_k$ on signal-flow diagrams. Understood in this light, the black-box functor $\blacksquare\colon \mathtt{SigFlow}_k \to \mathtt{FinRel}_k$ assigns to each signal-flow diagram its meaning: the linear relation it stands for.

Because controllability and observability involve extending k to $k(s)$, we will be concerned with the PROPs $\mathtt{FinRel}_{k(s)}$ and $\mathtt{SigFlow}_{k,s}$, where $\mathtt{SigFlow}_{k,s}$ is the free PROP $\mathtt{F}(\Sigma_{\mathtt{FinRel}_k,s})$, and $\Sigma_{\mathtt{FinRel}_k,s} = \Sigma_{\mathtt{FinRel}_k} \cup \{\sigma_{\int}\colon 1 \to 1\}$. We take the $\mathtt{F}$-image of $\sigma_{\int}$ to be a scaling by $\frac{1}{s}$ in the field extension $k(s)$. Then $k \cup \frac{1}{s}$ is a subset of $k(s)$, and we identify $\mathtt{SigFlow}_{k,s}$ with $\mathtt{SigFlow}_{k\cup\frac{1}{s}}$. Since $k(s)$ is generated as a field by $k \cup \frac{1}{s}$, the restriction of $\blacksquare\colon \mathtt{SigFlow}_{k(s)} \to \mathtt{FinRel}_{k(s)}$ to $\blacksquare\colon \mathtt{SigFlow}k,s \to \mathtt{FinRel}_{k(s)}$ is still full. A factor of $\frac{1}{s}$ comes from the Laplace transform of $\int_0^t f(\tau)d\tau$, so a 'scale by $\frac{1}{s}$' morphism will be referred to as an 'integrator'.

In order to extend the controllability and observability results in $\mathtt{Stateful}_k$ to $\mathtt{SigFlow}_{k,s}$, we need to find a subPROP $\mathtt{P}$ of $\mathtt{SigFlow}_{k,s}$ that maps to $\mathtt{Stateful}_k$ such that there are arrows making this diagram commute:

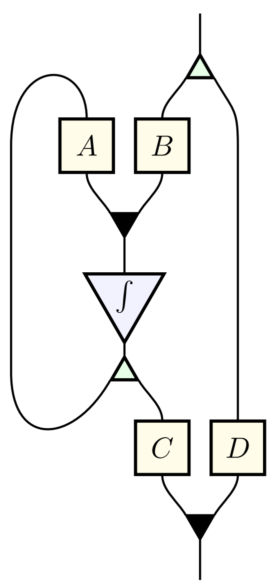
$$\begin{tikzcd}
\mathtt{P} \arrow[r] \arrow[d, hook, "j"'] & \mathtt{Stateful}_k \arrow[r, "\mathrm{eval}"] & \mathtt{FinVect}_{k(s)} \arrow[d, hook, "i"] \\
\mathtt{SigFlow}_{k,s} \arrow[rr, "\blacksquare"'] & & \mathtt{FinRel}_{k(s)}
\end{tikzcd},$$

where the evaluation map eval: $\mathtt{Stateful}_k \to \mathtt{FinVect}_{k(s)}$ mentioned in Theorem 18 sends any stateful morphism to the linear map it describes. Because $\mathtt{P}$ will be the PROP of signal-flow diagrams one might expect a control theorist to draw, we will name this PROP $\mathtt{ContFlow}_k$. Our goal, then, is to find this subPROP $\mathtt{ContFlow}_k$ of $\mathtt{SigFlow}_{k,s}$ and a PROP morphism $\Diamond\colon \mathtt{ContFlow}_k \to \mathtt{Stateful}_k$ such that

$$\begin{tikzcd}
\mathtt{ContFlow}_k \arrow[r, "\Diamond"] \arrow[d, hook, "j"'] & \mathtt{Stateful}_k \arrow[d, "i \circ \mathrm{eval}"] \\
\mathtt{SigFlow}_{k,s} \arrow[r, "\blacksquare"'] & \mathtt{FinRel}_{k(s)}
\end{tikzcd}$$

commutes.

This commutative square is not a pullback square, so $\mathtt{ContFlow}_k$ and $\Diamond$ cannot be simply defined by a pullback. A pullback square here would not give us a subPROP of $\mathtt{SigFlow}_{k,s}$, since $i \circ \mathrm{eval}$ is not a monomorphism in PROP. To define $\mathtt{ContFlow}_k$, we first need four processes that can be applied to any signal-flow diagram $f\colon m \to p$ in $\mathtt{SigFlow}_{k,s}$. In Chapter 4 we saw that signal-flow diagrams of the following form play a fundamental role in the state-space approach:



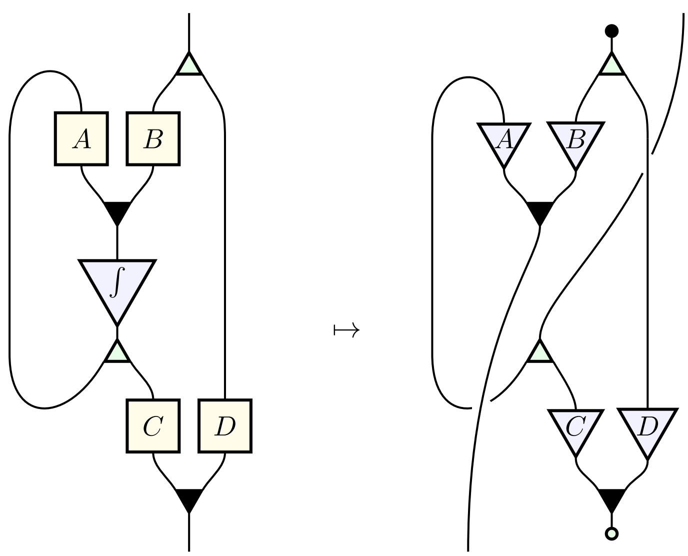
.

When applied to a signal-flow diagram of this form, our four processes pick out the linear relations A, B, C, and D — at least when they are linear *maps*. When all four of these linear relations are linear maps, we decree that f is a morphism in $\mathtt{ContFlow}_k$ and these linear maps form the $\mathtt{Stateful}_k$ morphism $\Diamond(f)$.

In what follows, we describe each of these four processes in generality and illustrate how they work for signal-flow diagrams of the above form. We choose the designations of m, n and p for the number of input wires, integrators, and output wires, respectively, on f in order to be consistent with the convention established in Section 4.3.

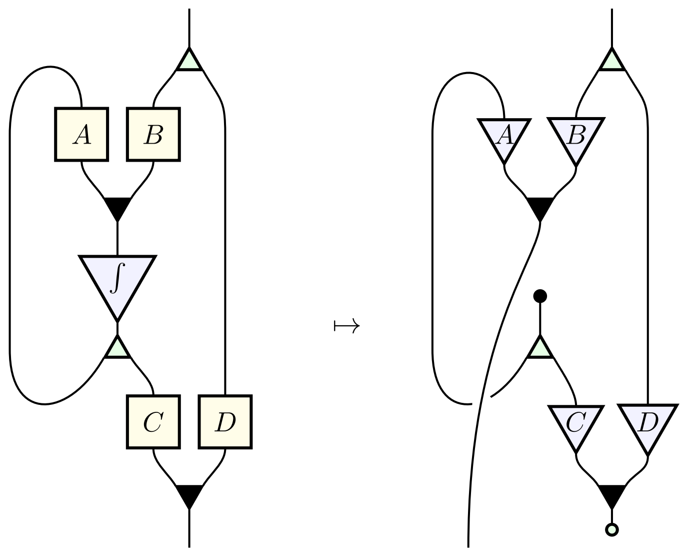
Definition 23 *The linear relation $A(f)\colon k^n \nrightarrow k^n$ is obtained from the signal-flow diagram f by replacing the n wires leaving the integrators in $!^p \circ f \circ 0^m$ with inputs and the n wires entering the integrators with outputs, then black-boxing the resulting signal-flow diagram.*

Example 24



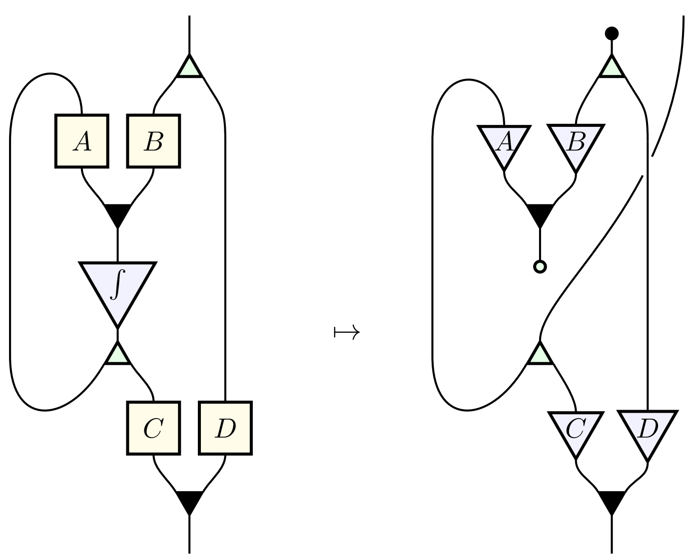
Definition 25 *The linear relation $B(f)\colon k^m \nrightarrow k^n$ is obtained from f by replacing the n wires entering the integrators in $!^p \circ f$ with outputs and replacing the n wires leaving the integrators with 0^n, then black-boxing.*

Example 26



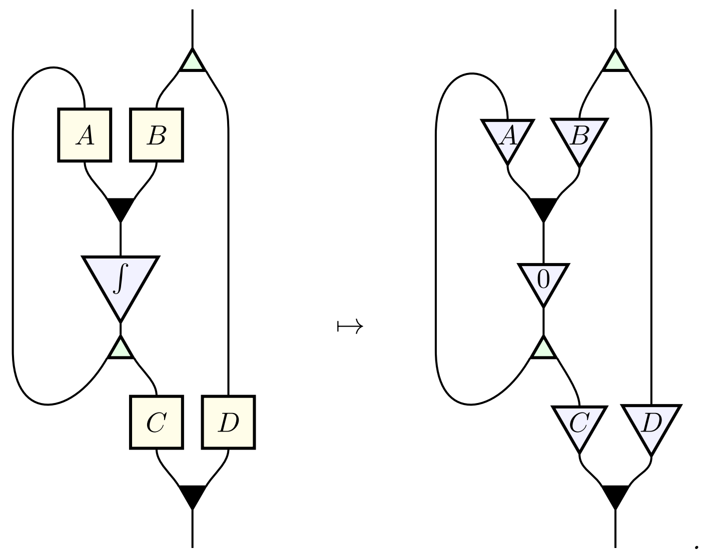
Definition 27 *The linear relation $C(f)\colon k^n \nrightarrow k^p$ is obtained from f dually, by replacing the n wires leaving the integrators in $f \circ 0^m$ with inputs and replacing the n wires entering the integrators with $!^n$, then black-boxing.*

Example 28



Definition 29 *The linear relation $D(f)\colon k^m \nrightarrow k^p$ is obtained by replacing each integrator with scaling by zero, then black-boxing.*

Example 30



Note, in addition to its connection to the state equations, the signal-flow diagram used for the examples is idealized to give a visual intuition of what the process does, with the design to suggest $A(f) = A$, $B(f) = B$, $C(f) = C$, and $D(f) = D$. As we shall see, this will be the case when $A(f)$, $B(f)$, $C(f)$, and $D(f)$ are all linear maps, deepening the connection between these processes and $\mathtt{Stateful}_k$, and hinting toward what makes a signal-flow diagram 'good'.

Definition 31 *The category $\mathtt{ContFlow}_{\boldsymbol{k}}$ is a subPROP of $\mathtt{SigFlow}_{k,s}$: a morphism $f\colon m \to p$ in $\mathtt{ContFlow}_k$ is a morphism $f\colon m \to p$ in $\mathtt{SigFlow}_{k,s}$ such that the linear relations $A(f)$, $B(f)$, $C(f)$, and $D(f)$ defined above are all linear maps.*

It is easy to see that all identity signal-flow diagrams are in $\mathtt{ContFlow}_k$. It is also clear that $\mathtt{ContFlow}_k$ is closed under direct sum. It is not as obvious that all composites of signal-flow diagrams in $\mathtt{ContFlow}_k$ are also in $\mathtt{ContFlow}_k$, but we show this in the proof of

Theorem 36, so $\mathtt{ContFlow}_k$ is indeed a PROP.

While this definition does not make it clear that the morphisms of $\mathtt{ContFlow}_k$ are closed under composition, it has some advantages over other definitions that appear reasonable, such as one that only insists 'there is a morphism $g\colon m \to p$ in $\mathtt{Stateful}_k$ with $i(\mathrm{eval}(g)) = \blacksquare(f)$'. While closure under composition is then immediate, this alternative definition does not guarantee the uniqueness of the morphism g. We can impose some extra conditions, for example requiring the number of integrators in f to be equal to the dimension of the state space of g, but even this fails to make g unique. For example, these morphisms in $\mathtt{Stateful}_k$ are different:

$$\begin{tikzcd} k \arrow[r, "\left[\frac{1}{s-1}\right]"] & k \arrow[d, "{[1]}"] \\ k \arrow[u, "{[0]}"] \arrow[r, "{[1]}"'] & k \end{tikzcd} \quad \neq \quad \begin{tikzcd} k \arrow[r, "\left[\frac{1}{s-1}\right]"] & k \arrow[d, "{[0]}"] \\ k \arrow[u, "{[1]}"] \arrow[r, "{[1]}"'] & k \end{tikzcd} ,$$

since the former is observable and the latter is not observable[1]. However, both have state spaces of dimension 1, and both evaluate to the identity relation, 1. As we can see from this example, we need to be intelligent in how we translate $\mathtt{ContFlow}_k$ signal-flow diagrams to $\mathtt{Stateful}_k$ morphisms if we want to have reasonable notions of controllability and observability. This will include knowing the number of integrators used, which explains why the category of signal-flow diagrams from which we get $\mathtt{ContFlow}_k$ needs to be $\mathtt{SigFlow}_{k,s}$ rather than $\mathtt{SigFlow}_{k(s)}$. This is reflected in the processes $A(f)$, $B(f)$, $C(f)$, and $D(f)$, since they cannot be defined on $\mathtt{SigFlow}_{k(s)}$.

While there is no reason *a priori* to expect a signal-flow diagram to be in such a convenient form as used in the examples above, the processes defined are unaffected by rewrites using the equations of $\mathtt{FinRel}_k$, where the integrators are left to be free. We therefore also consider the PROP $\mathtt{StFlow}_k = \mathtt{P}(\Sigma_{\mathtt{FinRel}_k,s}, E_{\mathtt{FinRel}_k})$ and its associated 'black-box'

[1]These two stateful morphisms can also be distinguished based on controllabilty: the former is not controllable, while the latter is controllable

functor $\blacklozenge\colon \texttt{SigFlow}_{k,s} \to \texttt{StFlow}_k$.

Lemma 32 *If two signal-flow diagrams f_1 and f_2 are in $\texttt{SigFlow}_{k,s}$ and $\blacklozenge f_1 = \blacklozenge f_2$, then $A(f_1) = A(f_2)$, $B(f_1) = B(f_2)$, $C(f_1) = C(f_2)$, and $D(f_1) = D(f_2)$.*

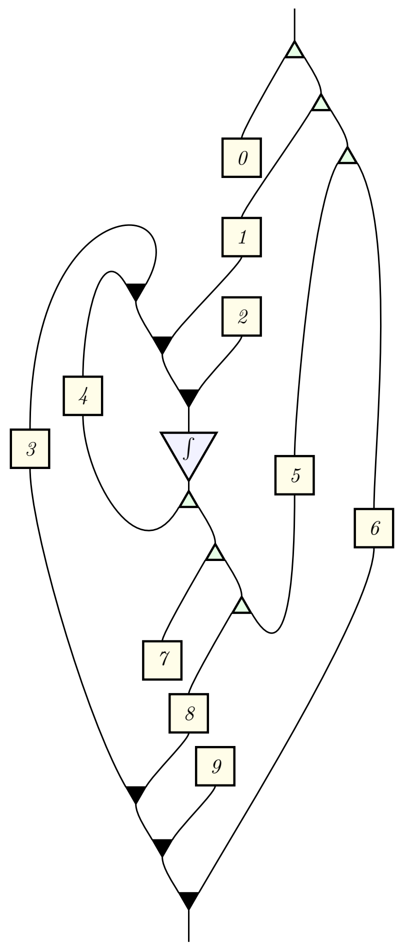
That is, rewrites from $\texttt{FinRel}_k$ (the ones that treat $\frac{1}{s}$ as a free generator) have no effect on $A(f)$, $B(f)$, $C(f)$, and $D(f)$.

Proof. We take advantage of the compositionality of signal-flow diagrams: a rewrite of $f \in \texttt{SigFlow}_{k,s}$ using one of equations **(1)–(31)** can be localized to a subdiagram of f that has no integrators in it. Doing a rewrite on such a subdiagram and then composing $!^p \circ f$ or $f \circ 0^m$ yields the same signal-flow diagram as first composing, then doing the same rewrite on that subdiagram. Likewise, rewrites of such subdiagrams and the changes to the integrators that occur in processes $A(f), \ldots, D(f)$ can be done in either order with the result of the same signal-flow diagram either way. Thus these rewrites 'commute' with the compositions and integrator replacements involved in these processes. Since black-boxing coequalizes signal-flow diagrams that differ in a rewrite from $\texttt{FinRel}_k$, a rewrite of f using one of equations **(1)–(31)** will have no effect on $A(f), \ldots, D(f)$.

Since any rewrite of f from $\texttt{FinRel}_k$ is a series of rewrites using equations **(1)–(31)**, no rewrite of f from $\texttt{FinRel}_k$ has an effect on $A(f), \ldots, D(f)$. ■

Up to this point we have focused on breaking up signal-flow diagrams into four subdiagrams via the PROP $\texttt{StFlow}_k$. The following lemma shows that, in terms of $\texttt{StFlow}_k$, there are actually ten subdiagrams than need to be considered in a fully general signal-flow diagram. The processes $A(f), \ldots, D(f)$ are each affected by several of these subdiagrams. Our next goal is to show that for a signal-flow diagram in $\texttt{ContFlow}_k$, six of these subdiagrams will be trivial, leaving only one non-trivial subdiagram each that can affect $A(f), \ldots, D(f)$.

Lemma 33 *Any morphism in $\texttt{StFlow}_k$ can be rewritten in the following form:*



,

where the numbered boxes are linear relations in $\mathtt{FinRel}_k$.

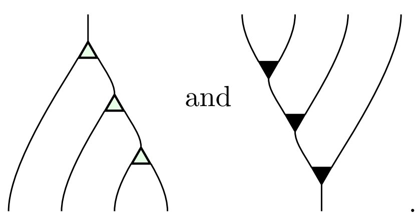
We will refer to this form as $\mathtt{StFlow}_k$-normal form.

Conjecture 34 *We can further take boxes 1,* $4^\dagger$*, 6, and 8 to be linear maps.*

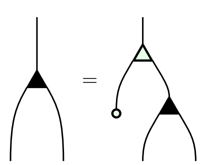
This conjecture is not necessary to our argument, but it would simplify future work extending our results if it is true.

We see there are two more ways to connect integrators, inputs, and outputs in $\mathtt{StFlow}_k$ that are not accounted for by A, B, C, and D (boxes 3 and 5), along with four self-connections (boxes 0, 2, 7, and 9). The input of an integrator can connect to an output (box 3), and an input can connect to the output of an integrator (box 5). When either of these kinds of connection occurs non-trivially in the signal-flow diagram f, $A(f)$ and $D(f)$ will not be linear maps, so these kinds of connection are trivial in any signal-flow diagram in $\mathtt{ContFlow}_k$. When any of the self-connections is non-trivial, either $A(f)$ or $D(f)$ will not be a linear map, so self-connections are also trivial for any signal-flow diagram in $\mathtt{ContFlow}_k$. Precisely what is meant by 'trivial' here is formalized in Lemma 35. Note: the status of $B(f)$ and $C(f)$ as linear maps is also affected by these extra connections.

Proof of Lemma 33. Since integrators are free in $\mathtt{StFlow}_k$, their placement in a diagram is important, relative to how their inputs and outputs connect to other parts of the diagram. Considering all the ways inputs, outputs, integrator inputs, and integrator outputs can connect, the diagram in Lemma 33 has all such possible connections in it. The only question, then, is whether there are other ways to join the connections that cannot be rewritten in this form. The joints here are all series of parallel Δ morphisms and series of parallel $+$ morphisms,

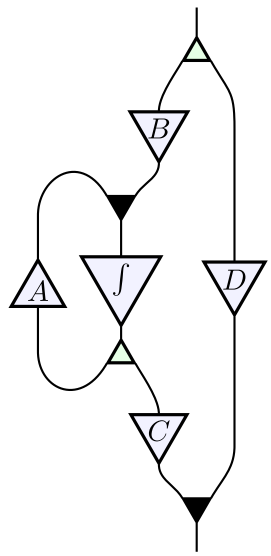


If a joint has a $+^\dagger$ in it, the input string to the $+^\dagger$ can be replaced with a $\Delta/!$ pair, as in equation **(4)**.



At this point, equation **(D5)**† can be applied as often as necessary to allow the $+^\dagger$s to assimilate into one or more of the boxes. Similarly, a joint with a $\Delta^\dagger$ in it can be transformed to a joint with only $+$ morphisms in it. ■

The next lemma shows when f is a signal-flow diagram in $\texttt{ContFlow}_k$, $\blacklozenge f$ can be written in the form



.

This means when $f \in \texttt{ContFlow}_k$, f can be transformed into the form of the signal-flow diagram used to demonstrate the processes $A(f), \ldots, D(f)$ without affecting $A(f), \ldots, D(f)$.

Lemma 35 *If $A(f), \ldots, D(f)$ are all linear maps, the corresponding boxes in the $\texttt{StFlow}_k$-normal form are these maps, and the other boxes are trivial. That is, box 0 is $!^m$, box 1 is $B(f)$, box 2 is 0^n, box 3 is $0^p \circ (0^\dagger)^n$, box 4 is $A(f)^\dagger$, box 5 is $(!^\dagger)^n \circ !^m$, box 6 is $D(f)$, box 7 is $(!^\dagger)^n$, box 8 is $C(f)$, and box 9 is 0^p.*

Proof. Applying process D to the $\texttt{StFlow}_k$-normal form, boxes 2, 4, and 7 become disconnected from the inputs and outputs. The portion of the diagram that remains can be rewritten in the form below, noted as diagram for D.

Similarly, the diagram for A is the result of applying process A to the $\texttt{StFlow}_k$-normal form and rewriting. The application of process A disconnects boxes 0, 6, and 9 from the inputs and outputs. The diagram for A is of the same form as the diagram for D, so we can apply the same arguments to corresponding linear relations.

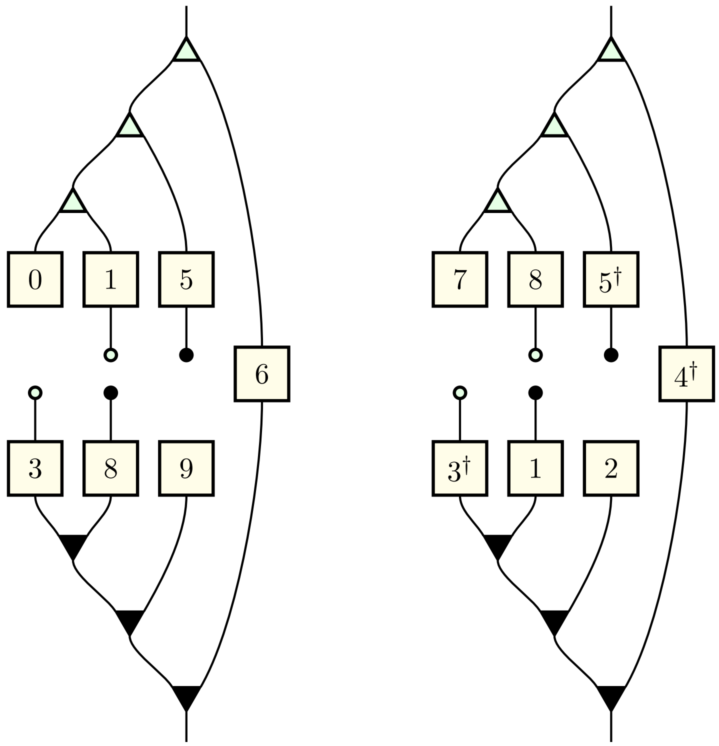
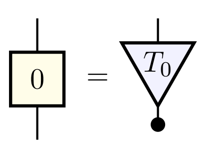


Diagram for D Diagram for A

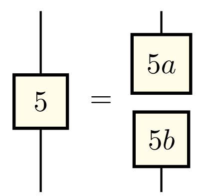
Assuming $A(f)$ and $D(f)$ to be linear maps, we first argue that the linear relations in boxes 0 and 7 must be trivial. Since the argument is the same for both, we focus on box 0.

Box 0 is a linear relation, which means it can be written in the standard form for linear relations that was demonstrated in Chapter 3. Box 0 also has no output, so it is a composite of a linear map T_0 and $(0^\dagger)^j$ for some $j \in \mathbb{N}$. If $j = 0$, T_0 has no outputs, meaning box 0 is $!^m$. If $j > 0$, one of the inputs of box 0 is a linear combination of the other inputs of box 0. Since the input to the diagram for D is duplicated to give the input to box 0, one of the inputs of $D(f)$ is a linear combination of the other inputs of $D(f)$. This means $D(f)$ is not a linear map, a contradiction. Thus the linear relation in box 0 is $!^m$, and the linear relation in box 7 must similarly be $!^n$.



The argument for boxes 2 and 9 is $*$-dual to the argument for boxes 0 and 7. The $*$-dual of the standard form for linear relations is an alternate standard form for linear relations, so each step of the above argument has a valid $*$-dual step. This means the linear relations in boxes 2 and 9 must be 0^n and 0^p, respectively.

Box 5 must have its inputs and outputs disconnected from each other, or else there is a non-trivial linear combination of the inputs of $D(f)$ that is equal to zero. This contradicts the assumption that $D(f)$ is a linear map.



Now the linear relation in box 5a is $!^m$ for the same reason as box 0, and the linear relation in box $(5\text{b})^\dagger$ is $!^n$ for the same reason as box 7. Thus the linear relation in box 5 must be $(!^\dagger)^n \circ !^m$.

Dually, the linear relation in box 3 must be $0^p \circ (0^\dagger)^n$.

If we can show boxes 1 and 8 are linear maps, that will force the linear relation in box 6 to be the only contribution to $D(f)$ and the $\dagger$-dual of the linear relation in box 4 to be the only contribution to $A(f)$, making box 6 a linear map and box $4^\dagger$ a linear map. With boxes 1, $4^\dagger$, 6, and 8 all linear maps and all other boxes trivial, the linear maps $B(f)$ and $C(f)$ must be boxes 1 and 8, respectively.

Assume, by way of contradiction, that box 1 is not a linear map. It is a linear relation, so one or both of the following must happen: some input to box 1 is a linear combination of the other inputs to box 1 or some input to box 1^* is a linear combination of the other inputs to box 1^*. The latter possibility is equivalent to some output of box 1 can take on multiple values given a fixed input to box 1. Since the inputs to $D(f)$ are duplicated to form the inputs to box 1, any linear dependence of the inputs to box 1 will translate into linear dependence of the same inputs to $D(f)$. This contradicts the assumption that $D(f)$ is a linear map.

Similarly, any linear dependence of the inputs to box 1^* will translate into linear dependence of the inputs to $A(f)^*$. Since $A(f)$ is assumed to be a linear map, it follows that $A(f)^*$ must also be a linear map, specifically the transpose map. Thus we have our contradiction for the latter case. This means the linear relation in box 1 must be a linear map. Swapping the roles of $A(f)$ and $D(f)$, we see the same must be true of box 8. ■

Aside from their not conforming to $\mathtt{ContFlow}_k$, there is a good control theory reason for why boxes 3 and 5 should be trivial. According to control theory folklore, a control system with differentiators in it will not be causal: the present state depends on future states and inputs. If this were not enough, if box 3 or box 5 is allowed to be non-trivial in $\mathtt{ContFlow}_k$, the PROP morphism analogous to $\Diamond$ in the next theorem no longer makes the commutative square commute.

Theorem 36 *There is a functor from* $\mathtt{ContFlow}_k$ *to* $\mathtt{Stateful}_k$ *given by*

$$\Diamond\colon \mathtt{ContFlow}_k \to \mathtt{Stateful}_k$$
$$f \mapsto (D(f), C(f), (sI - A(f))^{-1}, B(f)).$$

This functor is a PROP *morphism that makes the following diagram in* PROP *commute:*

$$\begin{tikzcd}
\mathtt{ContFlow}_k \arrow[r, "\Diamond"] \arrow[d, hook, "j"'] & \mathtt{Stateful}_k \arrow[d, "i \circ \mathrm{eval}"] \\
\mathtt{SigFlow}_{k,s} \arrow[r, "\blacksquare"'] & \mathtt{FinRel}_{k(s)}
\end{tikzcd}.$$

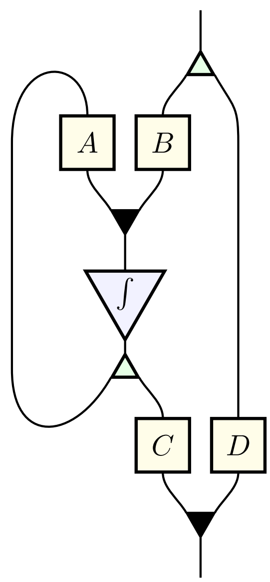
This functor is the means by which we translate the controllability and observability results from $\mathtt{Stateful}_k$ to $\mathtt{ContFlow}_k$.

Proof. To show $\Diamond$ is a functor, we need to check identities map to identities, and $\Diamond(f \circ f') = \Diamond(f) \circ \Diamond(f')$. The former is immediate, so we will focus on the latter.

To show $\lozenge$ is also a PROP morphism, we further need the distinguished object to map to the distinguished object, and $\lozenge(f \oplus f') = \lozenge(f) \oplus \lozenge(f')$. These additional criteria are straightforward to check, so we leave it to the reader to check them. To show the diagram in PROP commutes, we need for an arbitrary signal-flow diagram $f \in \mathtt{ContFlow}_k$ to satisfy $\blacksquare f = D(f) + C(f)(sI - A(f))^{-1}B(f)$.

- $\blacksquare f = D(f) + C(f)(sI - A(f))^{-1}B(f)$.

  By Lemma 35, any signal-flow diagram $f \in \mathtt{ContFlow}_k$ can be rewritten using the equations of $\mathtt{FinRel}_k$ into the form

  ,

  and this rewriting has no effect on $A(f), \ldots, D(f)$. Since the equations of $\mathtt{FinRel}_k$ are a subset of the equations of $\mathtt{FinRel}_{k(s)}$, the PROP morphism $\blacksquare$ factors through $\blacklozenge$. This means the rewriting will also have no effect on $\blacksquare f$, which imposes the equations of $\mathtt{FinRel}_{k(s)}$ onto f. By imposing all of the equations of $\mathtt{FinRel}_{k(s)}$ on this diagram, we get $\blacksquare f = D + C(sI - A)^{-1}B$. By applying the processes $A, \ldots, D$ to this diagram, we see that $A(f) = A$, $B(f) = B$, $C(f) = C$, and $D(f) = D$.

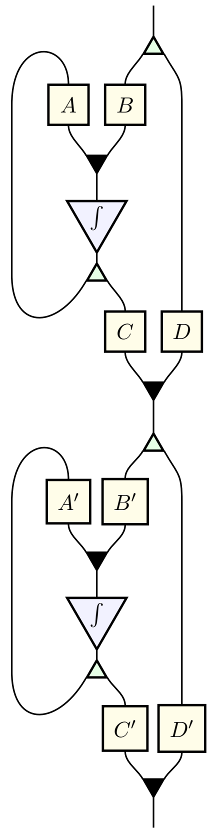
- $\lozenge(f \circ f') = \lozenge(f) \circ \lozenge(f')$.

Suppose f and f' are signal-flow diagrams in $\texttt{ContFlow}_k$ which are composable in $\texttt{SigFlow}_{k,s}$, with $\Diamond(f) = (D, C, (sI - A)^{-1}, B)$ and $\Diamond(f') = (D', C', (sI' - A')^{-1}, B')$. In Chapter 4 we saw how to compose two stateful morphisms, $\Diamond(f) \circ \Diamond(f')$, so we need to check that we get the same composite from $\Diamond(f \circ f') = (D'', C'', (sI'' - A''), B'')$. This would mean $A'', \ldots, D''$ are all linear maps, finally justifying the assertion above that composition is closed in $\texttt{ContFlow}_k$. In matrix form, that means we need to verify:

$$D'' = DD' \qquad\qquad C'' = \begin{bmatrix} D'C & C' \end{bmatrix}$$

$$B'' = \begin{bmatrix} B \\ B'D \end{bmatrix} \qquad\qquad A'' = \begin{bmatrix} A & 0 \\ B'C & A' \end{bmatrix}.$$

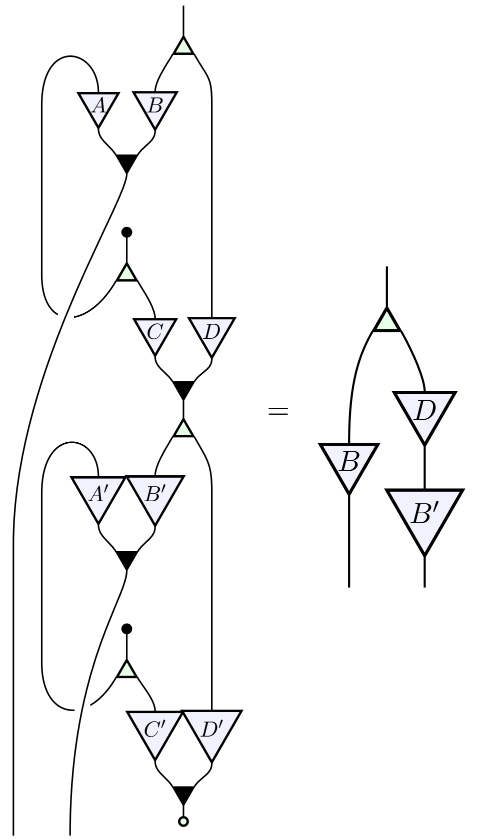
We will be performing several surgeries on the signal-flow diagram $f \circ f'$, so let's take a good look at the 'patient'. By Lemma 35, we need only consider a signal-flow diagram of the form:



.

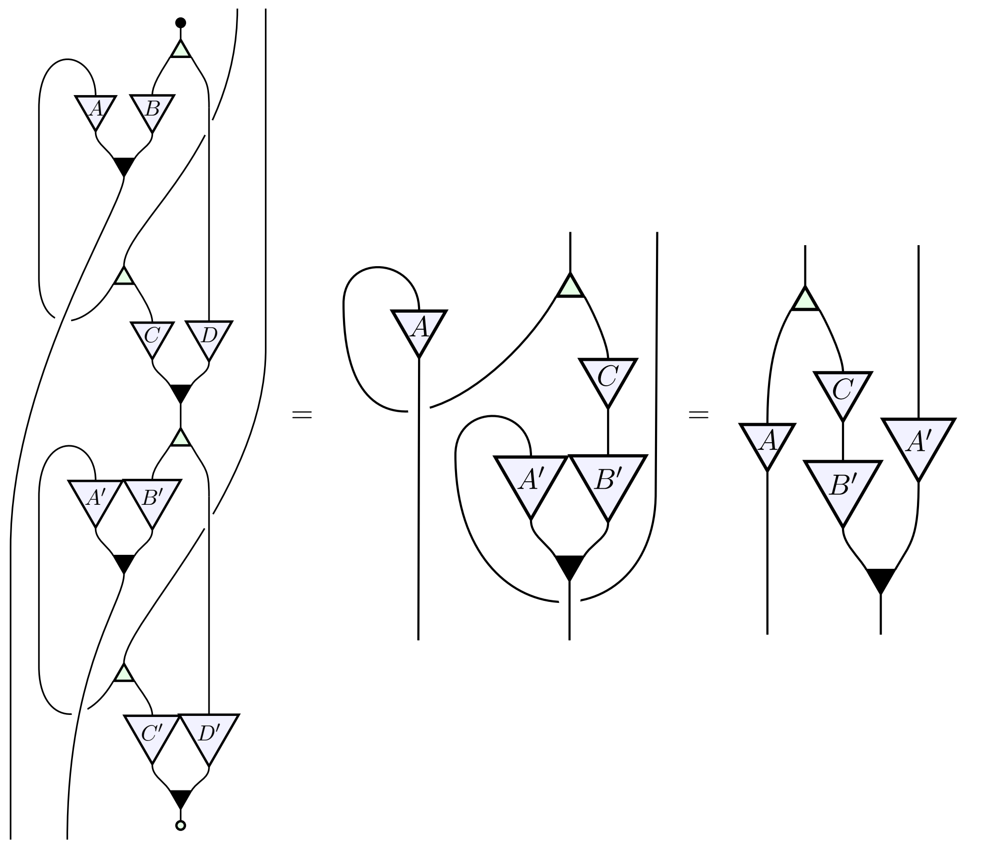
Since the $D(f)$ part of the process is functorial, $D'' = D(f \circ f') = D(f) \circ D(f') = DD'$. The $B(f)$ and $C(f)$ parts are dual to each other, so we show only the argument for B'' and leave C'' as an easy exercise.

To find $B'' = B(f \circ f')$, we replace the $n+n'$ wires entering the integrators in $!^p \circ f \circ f'$ with outputs and replace the $n+n'$ wires leaving the integrators with $0^{n+n'}$. Since the 0s always meet linear maps, they destroy everything in their paths until they reach a $+$, at which point equation **(1)** gives an identity wire. This gets rid of A, A', C, C', and all the $+$s. Similarly, each ! will always meet linear maps, destroying everything in their paths until they reach a Δ, at which point equation **(4)** gives an identity wire. This gets rid of D' and would get rid of C' if it were not already gone. From these considerations, we get the following results on our 'patient':



That is, $B'' = \begin{bmatrix} B \\ B'D \end{bmatrix}$, which is exactly what is required. As noted above, the argument for C'' is just a dual version of this one, reflected about the x-axis and with colors swapped.

To find $A'' = A(f \circ f')$, we replace the $n+n'$ wires leaving the integrators in $!^p \circ f \circ f' \circ 0^{m'}$ with inputs and the $n+n'$ wires entering the integrators with outputs. As with the 0s in B'', the 0s here always meet linear maps, so they destroy everything in their paths until they reach a $+$, and likewise for the !s until they reach a Δ. This time the zeros get rid of B and D, while the deletions get rid of C' and D'. From there some zig-zags can be straightened out, and finally commutativity allows A' and B' to be swapped. Graphically, this proceeds as follows:



This final diagram for A'', written in block matrix form is $\begin{bmatrix} A & 0 \\ B'C & A' \end{bmatrix}$.

This is exactly what was required. ■

5.2 Duality properties of $\texttt{ContFlow}_k$

Recall that $\texttt{SigFlow}_k$ has two different dagger structures, † and ∗. As noted in Section 4.4, controllability and observability are dual concepts, with the duality relating to transposition. Since $-^*$ is also a duality related to transposition, this suggests a connection between the controllable/observable duality and the $-^*$ duality.

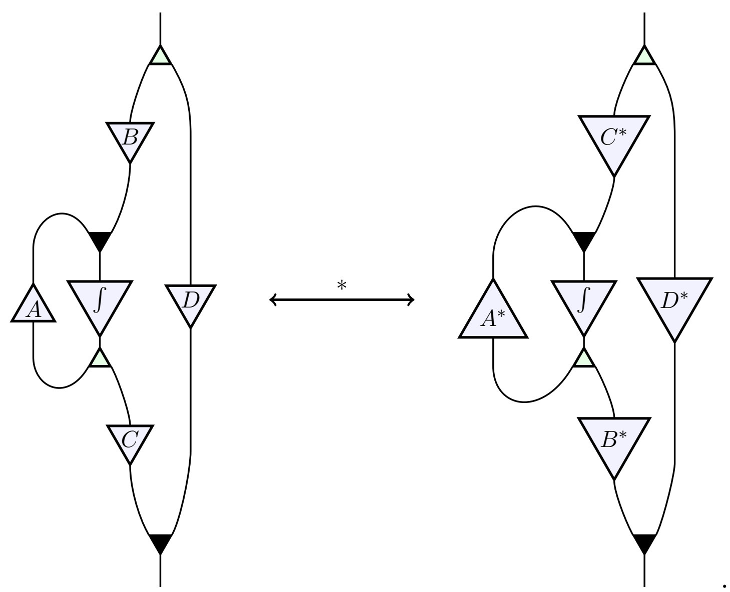
Proposition 37 $\texttt{ContFlow}_k$ *is a dagger-category in only one of the two ways that* $\texttt{SigFlow}_{k,s}$ *is. Specifically, the* $-^*$ *dual of a morphism* f *in* $\texttt{ContFlow}_k$ *is again a morphism in*

$\texttt{ContFlow}_k$ *such that*

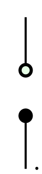
$$A(f^*) = A(f)^* \qquad B(f^*) = C(f)^*$$

$$D(f^*) = D(f)^* \qquad C(f^*) = B(f)^*.$$

Proof. That the $-^*$ duality behaves as described is an immediate consequence of $\blacklozenge(f^*) = \blacklozenge(f)^*$. Thus we have



To show $\texttt{ContFlow}_k$ does not have the $-^\dagger$ duality, it suffices to give a counterexample. Let f be any signal-flow diagram such that $D(f) = 0\colon k \to k$. For example take f to be the signal-flow diagram:



We see $D(f^\dagger)$ is the scaling by 0^{-1}. While this is a linear relation, it is clearly not a linear map. ∎

Our signal-flow diagrams have all been time-independent, so the time reversal part in Kalman's duality is trivial for our diagrams. Note also that the $-^*$ duality is exactly transposition when applied to linear maps, so we can rewrite the equations in Proposition 37:

$$A(f^*) = (A(f))^\top \qquad\qquad B(f^*) = (C(f))^\top$$
$$D(f^*) = (D(f))^\top \qquad\qquad C(f^*) = (B(f))^\top.$$

At the level of state-space equations, this is exactly the time-independent version of Kalman's duality! We predict the $-^*$ duality can be rigorously extended to time-dependent signal-flow diagrams in such a way that it exactly matches Kalman's duality. In the present work we have taken advantage of time-independence in the definition of $\mathtt{Stateful}_k$, restricting the appearance of Laplace transform variable s to the $(sI - A)^{-1}$ part. The time-dependent case will require a new approach to $\mathtt{Stateful}_k$, hence a new approach to $\mathtt{ContFlow}_k$.

Chapter 6

Conclusions

While the story of control theory placed into the context of category theory is far from complete, we have advanced the plot. First, in Chapter 3 we found a symmetric monoidal category that describes the relation between the inputs and outputs of signal-flow diagrams and described it in terms of generating morphisms and a set of equations between morphisms. The strict version of this gave us $\mathtt{FinRel}_k$. Bonchi, Sobociński and Zanasi [6, 7] independently studied an equivalent symmetric monoidal category around the same time and with the same generator-and-equations perspective, but from a very different approach and with a slightly different set of generators. On our way to $\mathtt{FinRel}_k$, we considered $\mathtt{FinVect}_k$, which is the strict symmetric monoidal category we would get if signal-flow diagrams had no feedback. Wadsley and Woods [35] considered $\mathtt{Mat}(k)$, which is equivalent to our $\mathtt{FinVect}_k$ when k is a field, but only insisted that k be a rig. There are also many mysterious and interesting connections between $\mathtt{FinRel}_k$ and symmetric monoidal categories used in quantum mechanics [1, 5, 10, 11, 12, 9, 20, 21, 27, 28, 34], exposing similarities and subtle differences between vector spaces, Hilbert spaces, and cobordisms. Nevertheless, $\mathtt{FinRel}_{k(s)}$ is helpless to describe certain important control theory concepts such as controllability and observability.

To get a handle on these two concepts, we went back in Chapter 4 to the state-space equations, Equations 1.1 and 1.2 upon which these concepts are founded. By encoding these equations in signal-flow diagrams, we found the category $\mathtt{Stateful}_k$, where the four matrices in the state-space equations taken *en masse* correspond to stateful morphisms. While for simplicity's sake we only considered the linear time-independent case where the four matrices are constant in time, $\mathtt{Stateful}_k$ can easily be extended to the time-varying case by adding a very mild condition on the matrices. Controllability and observability only depend on the matrices in the state-space equations in either case, so knowing a stateful morphism provides enough information to determine controllability and observability. The analysis simplifies greatly in the linear time-independent case, where they can be determined in terms of epimorphisms and monomorphisms.

In Chapter 5 we pushed the idea of controllability and observability a little further. Stateful morphisms can only be determined if we already know what the matrices in the state-space equations are, which leaves the issue of finding these matrices. Given an arbitrary signal-flow diagram, the values of these matrices may not be obvious, or worse, linear relations may be involved, not just linear maps. For this reason we limited the scope of signal-flow diagrams to form a new category $\mathtt{ContFlow}_k$, where only the signal-flow diagrams that can be converted to stateful morphisms are considered. These signal-flow diagrams coincide with the ones most typically drawn by control theorists. By converting a signal-flow diagram in $\mathtt{ContFlow}_k$ to a stateful morphism, we can determine controllability and observability for that signal-flow diagram.

While controllability and observability are important, there are many other concepts that are important to control theorists, such as stability (which itself comes in several guises) and pole placement. On the other hand, there are several opportunities for the current work to be extended to allow control theorists to draw more general signal-flow diagrams. Our $\mathtt{Stateful}_k$ PROP, which finally allowed us to describe controllability and observability in the category theory context, should be extendable using the results of Appendix B, allowing a larger collection of signal-flow diagrams to be considered for controllability and observability.

Once we have extended $\mathtt{Stateful}_k$ in this way, we can also extend $\mathtt{ContFlow}_k$. We have seen in Chapter 5 how the state-space equations 1.1 and 1.2 can be used to form the PROP $\mathtt{ContFlow}_k$. One of the features of $\mathtt{ContFlow}_k$ is the deterministic nature of its morphisms: the current state and input uniquely determine the future state and output. We may not be able to induce the system to enter a given state if the system is not controllable, and we may not be able to determine the state of the system if it is not observable, so the state of a system can act as a 'hidden variable'. However, in some circumstances it may be useful to eschew determinism. We can do this by generalizing the state-space *equations* to state-space *relations* in a way that does not sacrifice much of the convenience of dealing with linear maps.

To get a flavor of this, we will still insist B, C and D are linear maps, only allowing A to be replaced with a linear relation. In Appendix B we see that we can generalize a bit more than this, but the full generality at this point would only serve to weigh down the exposition, obscuring what we wish to point out: a direction for extending $\mathtt{ContFlow}_{\mathbb{R}}$. The state-space relations will then appear as:

$$\dot{x}(t) \in A(t)x(t) + B(t)u(t) \tag{6.1}$$

$$y(t) = C(t)x(t) + D(t)u(t). \tag{6.2}$$

Given two such systems: $\dot{x}_1 \in A_1x_1 + B_1u_1$, $y_1 = C_1x_1 + D_1u_1$ and $\dot{x}_2 \in A_2x_2 + B_2y_1$, $y_2 = C_2x_2+D_2y_1$; they can compose by writing all the relations and eliminating the common

y_1. Thus:

$$\dot{x}_1 \in A_1 x_1 + B_1 u_1$$
$$\dot{x}_2 \in A_2 x_2 + B_2(C_1 x_1 + D_1 u_1)$$
$$y_2 = C_2 x_2 + D_2(C_1 x_1 + D_1 u_1).$$

Since each B, C and D is a linear map, compositions of these linear relations still distribute over addition, so the composite system can be written

$$\dot{x}_1 \in A_1 x_1 + B_1 u_1$$
$$\dot{x}_2 \in (B_2 C_1)x_1 + A_2 x_2 + (B_2 D_1)u_1$$
$$y_2 = (D_2 C_1)x_1 + C_2 x_2 + (D_2 D_1)u_1.$$

It is easy to show this is equivalent to

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \in A_3 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 D_1 \end{bmatrix} u_1$$
$$y_2 = \begin{bmatrix} D_2 C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + (D_2 D_1)u_1,$$

for some linear relation A_3. The fact that we get a system of state-space relations again indicates that we should be able to form a PROP $\mathtt{CtrlFlow}_k$ from the state-space relations in much the way that we formed $\mathtt{ContFlow}_k$ from the state-space equations.

A difficulty in dealing with $\mathtt{CtrlFlow}_k$ arises when trying to determine the linear relations $A(f)$, $B(f)$, $C(f)$, and $D(f)$ associated with the signal-flow diagram $f \in \mathtt{CtrlFlow}_k$. The processes defined for finding linear maps from a signal-flow diagram $f \in \mathtt{ContFlow}_k$ are not appropriate here. When $A(f)$ is a linear relation that is not a linear map, these processes will give one or both of $B(f)$ and $C(f)$ as linear relations that are not linear maps. If Conjecture 34 is true, it may be possible to use the $\mathtt{StFlow}_k$-normal form of a signal-flow diagram to find the linear relations $A(f)$, $B(f)$, $C(f)$, and $D(f)$. This approach, if it works, would be more satisfying than the current *ad hoc* approach to finding these when they are linear maps.

Another direction for research, taken up by Baez, Coya and Rebro [2], is connecting this work with the work of Baez and Fong [4] on passive linear networks. Whereas we have primarily focused on presenting PROPs in terms of generators and equations, Baez and Fong use a framework of 'decorated cospans'. Using this framework, they find a black-box functor from Circ, the category of open passive linear electric circuits, to $\mathrm{LagrRel}_{k(s)}$, the category of symplectic vector spaces over the field $k(s)$ and Lagrangian relations. These categories are equivalent to their skeletons, so there is a black-box functor between their skeletons, $\blacksquare\colon \mathtt{Circ} \to \mathtt{LagrRel}_{k(s)}$. Since there is a symmetric monoidal dagger functor $i\colon \mathtt{LagrRel}_{k(s)} \to \mathtt{FinRel}_{k(s)}$ that includes $\mathtt{LagrRel}_{k(s)}$ in $\mathtt{FinRel}_{k(s)}$, composing with the black-box functor gives the symmetric monoidal dagger functor

$$i \circ \blacksquare\colon \mathtt{Circ} \to \mathtt{FinRel}_{k(s)}.$$

A key result here is to find a symmetric monoidal dagger functor $F\colon \mathtt{Circ} \to \mathtt{SigFlow}_{k(s)}$ such that the black-box functor we defined from $\mathtt{SigFlow}_{k(s)}$ to $\mathtt{FinRel}_{k(s)}$ makes this functor diagram commute (up to isomorphism):

$$\begin{tikzcd}
\mathtt{Circ} \arrow[r, "\blacksquare"] \arrow[d, "F"'] & \mathtt{LagrRel}_{k(s)} \arrow[d, hook, "i"] \\
\mathtt{SigFlow}_{k(s)} \arrow[r, "\blacksquare"'] & \mathtt{FinRel}_{k(s)}
\end{tikzcd}.$$

The main challenge in this endeavor is not in finding the functor F, but in reconciling the approaches sufficient to prove the square commutes.

Bibliography

[1] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LiCS'04)*, pages 415–425, July 2004. DOI 10.1109/LICS.2004.1319636.

[2] John Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. In preparation.

[3] John Baez and Jason Erbele. Categories in control. *Th. Appl. Cat.*, 30(24):836–881, 2015. Available at http://www.tac.mta.ca/tac/volumes/30/24/30-24abs.html.

[4] John Baez and Brendan Fong. A compositional framework for passive linear networks. *arXiv preprint*, 2015. Available at arXiv:1504.05625.

[5] John Baez and Mike Stay. Physics, topology, logic and computation: a Rosetta Stone. In Bob Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 95–172. Springer, 2011. Also available at arXiv:0903.0340.

[6] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014–Concurrency Theory*, Lecture Notes in Computer Science, pages 435–450. Springer, 2014. DOI 10.1007/978-3-662-44584-6_30.

[7] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Interacting Hopf algebras. *arXiv preprint*, 2014. Available at arXiv:1403.7048.

[8] Francis Borceux. *Handbook of Categorical Algebra. 2*, volume 51 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994. DOI 10.1017/CBO9780511525865.

[9] Aurelio Carboni and Robert F.C. Walters. Cartesian bicategories I. *J. Pure Appl. Alg.*, 49(1):11–32, 1987.

[10] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New J. Phys.*, 13(4):043016, 2011.

[11] Bob Coecke and Éric Oliver Paquette. Categories for the practising physicist. In Bob Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*,

pages 173–286. Springer, 2010. DOI 10.1007/978-3-642-12821-9_3. Also available at arXiv:0905.3010.

[12] Bob Coecke, Dusko Pavlovic, and Jamie Vicary. A new description of orthogonal bases. *Math. Str. Comp. Sci.*, 23(03):555–567, 2013. Also available at arXiv:0810.0812.

[13] Brendan Fong, Paolo Rapisarda, and Paweł Sobociński. A categorical approach to open and interconnected systems. *arXiv preprint*, 2015. Available at arXiv:1510.05076.

[14] Bernard Friedland. *Control System Design: An Introduction to State-Space Methods.* Courier Dover Publications, 2012.

[15] Chris Heunen and Jamie Vicary. *Introduction to Categorical Quantum Mechanics.* preprint, 2014.

[16] André Joyal and Ross Street. The geometry of tensor calculus I. *Adv. Math.*, 88(1):55–113, 1991. DOI 10.1016/0001-8708(91)90003-P.

[17] André Joyal and Ross Street. The geometry of tensor calculus II. Draft available at http://maths.mq.edu.au/~street/GTCII.pdf.

[18] Rudolf Emil Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.4070.

[19] Rudolf Emil Kalman. Mathematical description of linear dynamical systems. *J.S.I.A.M. Control, Ser. A*, 1(2):152–192, 1963. DOI 10.1137/0301010.

[20] Joachim Kock. *Frobenius Algebras and 2D Topological Quantum Field Theories*, volume 59 of *London Mathematical Society Student Texts.* Cambridge University Press, 2004. Short version available at http://mat.uab.es/~kock/TQFT/FS.pdf.

[21] Joachim Kock. Remarks on the origin of the Frobenius equation. Available at http://mat.uab.es/~kock/TQFT.html#history.

[22] Stephen Lack. Composing PROPs. *Th. Appl. Cat.*, 13(9):147–163, 2004. Available at http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html.

[23] Yves Lafont. Towards an algebraic theory of boolean circuits. *J. Pure Appl. Alg.*, 184(2):257–310, 2003. Also available at http://iml.univ-mrs.fr/~lafont/pub/circuits.pdf.

[24] F. William Lawvere. *Functorial Semantics of Algebraic Theories.* PhD thesis, Columbia University, 1963. Also available at http://www.tac.mta.ca/tac/reprints/articles/5/tr5abs.html.

[25] Saunders Mac Lane. Natural associativity and commutativity. *Rice U. Studies*, 49(4):28–46, 1963.

[26] Samuel Mimram. Presentation of a game semantics for first-order propositional logic. *arXiv preprint*, 2008. Available at arXiv:0805.0845.

[27] Robert Rosebrugh, Nicoletta Sabadini, and Robert F. C. Walters. Generic commutative separable algebras and cospans of graphs. *Th. Appl. Cat.*, 15(6):164–177, 2005. Available at http://www.tac.mta.ca/tac/volumes/15/6/15-06abs.html.

[28] Peter Selinger. Dagger compact closed categories and completely positive maps. *Elec. Notes Theor. Comp. Sci.*, 170:139–163, 2007.

[29] Paweł Sobociński. Copying, Discarding and The Slogan. *Graphical Linear Algebra*, May 9, 2015. Available at http://graphicallinearalgebra.net/2015/05/09/copying-discarding-and-the-slogan.

[30] Paweł Sobociński. Integer matrices. *Graphical Linear Algebra*, July 8, 2015. Available at http://graphicallinearalgebra.net/2015/07/08/integer-matrices.

[31] Paweł Sobociński. When Adding met Copying... *Graphical Linear Algebra*, May 12, 2015. Available at http://graphicallinearalgebra.net/2015/05/12/when-adding-met-copying.

[32] Paweł Sobociński. Dividing by zero to invert matrices. *Graphical Linear Algebra*, June 22, 2016. Available at http://graphicallinearalgebra.net2016/06/22/29-inverting-matrices-and-dividing-by-zero.

[33] Todd Trimble. Multisorted Lawvere theories. *nLab*, April 27, 2014. Available as https://ncatlab.org/toddtrimble/published/multisorted+Lawvere+theories.

[34] Jamie Vicary. Categorical formulations of finite-dimensional quantum algebras. *Comm. Math. Phys.*, 304(3):765–796, 2011. Also available at arXiv:0805.0432.

[35] Simon Wadsley and Nick Woods. PROPs for linear systems. *arXiv preprint*, 2015. Available at arXiv:1505.00048.

[36] Jan C. Willems. Models for dynamics. In U. Kirchgraber and H. O. Walther, editors, *Dynamics Reported*, volume 2, pages 171–269. John Wiley & Sons Ltd and B.G. Teubner, 1989. DOI 10.1007/978-3-322-96657-5_5. Also available at http://homes.esat.kuleuven.be/~jwillems/Articles/JournalArticles/1989.1.pdf.

[37] Fabio Zanasi. *Interacting Hopf Algebras—the Theory of Linear Systems.* PhD thesis, École Normale Supérieure de Lyon, 2015. Available at https://tel.archives-ouvertes.fr/tel-01218015/.

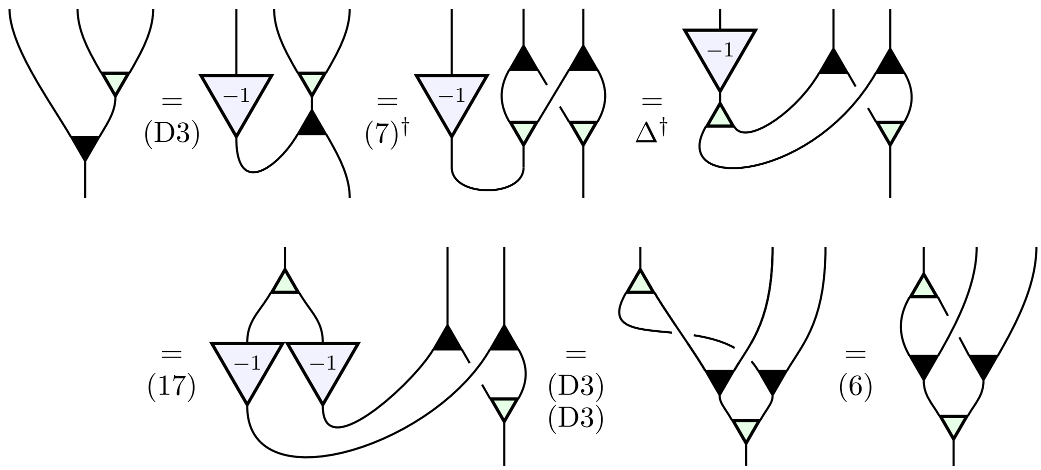
Appendix A

Proofs of selected derived equations

In the proof of Theorem 15 we used several equations derived from the equations in our presentation of $\mathtt{FinRel}_k$. While the more straightforward equations were demonstrated immediately, some of the more useful derived equations are less straightforward and we demonstrate them here. While these derived equations could simply be appended to our presentation, the elegance of only using simple structures would be lost. On the other hand, Heunen and Vicary [15] pointed out only one Frobenius equation (per color) is necessary instead of two. Several of the equations necessary for the presentation of $\mathtt{FinVect}_k$ are also superfluous for the presentation of $\mathtt{FinRel}_k$, but it seems more elegant to build from simple structures than to minimize the number of equations for the sake of minimization.

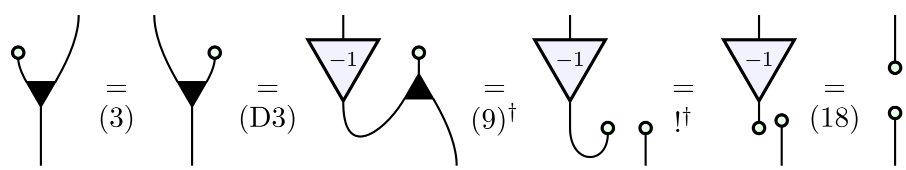
A.1 (D5)

Derived equations **(D5)–(D7)** are variations on the bimonoid equations **(7)–(9)**. Derived equation **(D5)** can be proved as follows:



A.2 (D6)–(D7)

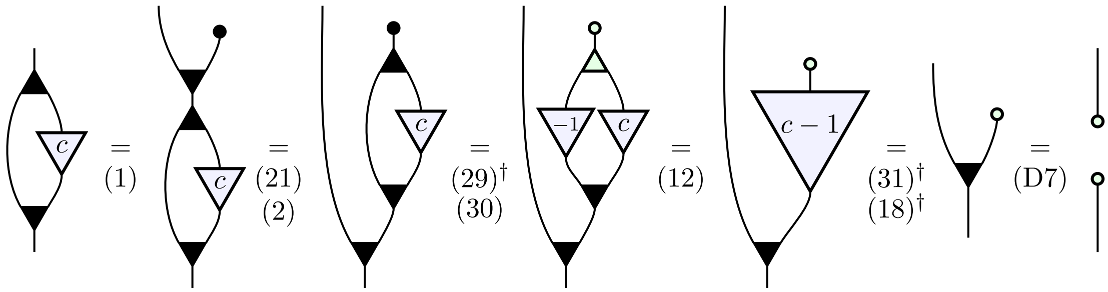
Derived equation **(D7)** can be proved as follows:



The proof of derived equation **(D6)** is a vertically flipped and color-swapped version of the proof of derived equation **(D7)** above, but without the scaling by -1. These two equations are also proved in the *Graphical Linear Algebra* blog.

A.3 (D8)–(D9)

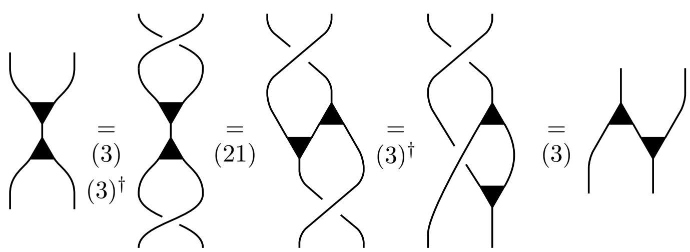
Derived equation **(D8)** is simply a statement that $x + y$ and $x + cy$ are linearly independent whenever $c \neq 1$. This can be proved diagrammatically as follows:



Derived equation **(D9)** is the statement that when $c \neq 1$, $x = cx$ implies $x = 0$. The proof of derived equation **(D9)** is a vertically flipped and color-swapped version of the proof of derived equation **(D8)** above.

A.4 Frobenius equations

In our presentation of $\mathtt{FinRel}_k$, two of the Frobenius equations are superfluous. For each pair of Frobenius equations **(21)–(22)** and **(23)–(24)** either equation can be derived from the other. Furthermore, both equations of a pair can be derived from the 'outer' equation. Here we show how to derive equation **(22)** from equation **(21)**, commutativity of $+$, and cocommutativity of $+^\dagger$.

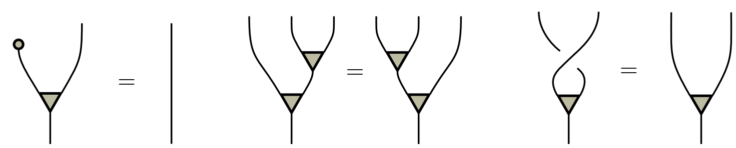


Color swapping this argument gives an argument for deriving equation **(24)** from equation **(23)**. One way to arrive at the numbered Frobenius equations from an outer equation uses counitality and coassociativity. Putting the pieces together for this is left as an exercise to the reader.

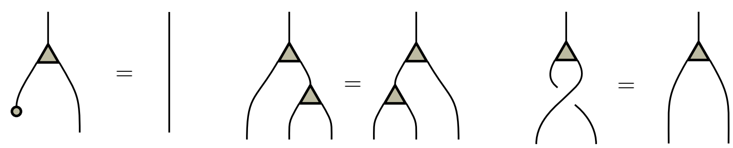
A.5 Additional connections

Recall that a special commutative Frobenius monoid is:

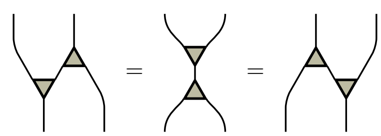
(F1)–(F3) a commutative monoid:



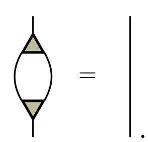
(F4)–(F6) which is also a cocommutative comonoid:



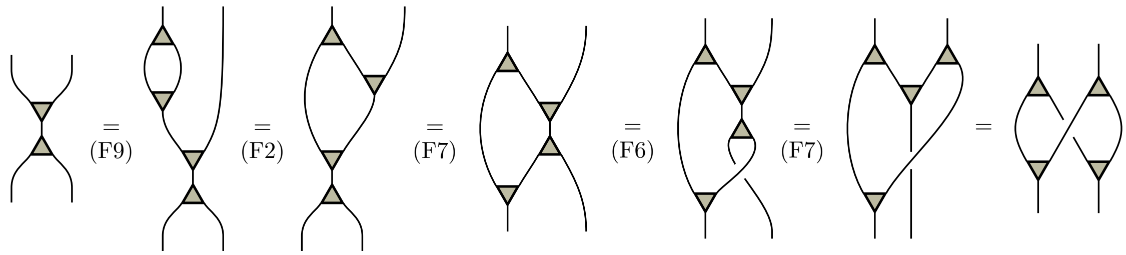
(F7)–(F8) that satisfies the Frobenius equations:



(F9) and the special equation:



We have seen one of the bimonoid laws, equation **(10)**, is compatible with the special commutative Frobenius monoid structure without making it trivial. Indeed, we get the extra-special commutative Frobenius monoid structure when we include this equation. We can see below that equation **(7)** is not only compatible with the special commutative Frobenius monoid structure, it is a derived equation under the assumption of a special commutative Frobenius monoid structure. As noted by Heunen and Vicary [15], if either of the other two bimonoid equations (**(8)–(9)**) hold for a Frobenius monoid, the monoid is trivial, so the characteristic difference between non-trivial bimonoids and non-trivial Frobenius monoids is the way the unit and counit interact with the comultiplication and multiplication, respectively.



The last equality is due to the naturality of symmetry.

Appendix B

Generalization of the Box construction

In Section 4.2 we described the Box construction in the particular case of $\mathcal{C} = \texttt{FinVect}_k$. The description does not change much when we allow $\mathcal{C}$ to be an arbitrary category with biproducts. A more substantial generalization is required in order to consider the case of $\mathcal{C} = \texttt{FinRel}_k$. We build up to the PROP $\square(\texttt{FinRel}_k)$ here, but leave its exploitation for future work.

Definition 38 *Given a category $(\mathcal{C}, m, 0, \Delta, !)$ with biproducts, we take $\square\mathcal{C}$ to have*

- *the same objects as $\mathcal{C}$,*
- *the morphisms in $\hom(X, Y)$ as equivalence classes of*

$$X \xrightarrow{\Delta} X \otimes X \xrightarrow{\mathrm{id}_X \otimes b} X \otimes S \xrightarrow{\mathrm{id}_X \otimes a} X \otimes T \xrightarrow{d \otimes c} Y \otimes Y \xrightarrow{m} Y\ ,$$

abbreviated as (d, c, a, b),

- *composition given by*

$$(d,c,a,b)\circ(d',c',a',b') = \left(d'd, [d'c \quad c'], \begin{bmatrix} a & 0 \\ a'b'ca & a' \end{bmatrix}, \begin{bmatrix} b \\ b'd \end{bmatrix}\right).$$

The morphisms in $\hom(X,Y)$ are more convenient to work with when depicted as non-commuting squares, as in

$$\begin{tikzcd} S \arrow[r, "a"] & T \arrow[d, "c"] \\ X \arrow[u, "b"] \arrow[r, "d"'] & Y \end{tikzcd}.$$

This form for morphisms explains the name $\square\mathcal{C}$.

Two squares, (d,c,a,b) and (d,c',a',b') are in the same equivalence class if there are isomorphisms $\alpha\colon S \to S'$ and $\omega\colon T \to T'$ such that the following diagram in $\mathcal{C}$ commutes:

$$\begin{tikzcd} S \arrow[rrr, "a"] \arrow[ddd, "\alpha"'] & & & T \arrow[dl, "c"'] \arrow[ddd, "\omega"] \\ & X \arrow[ul, "b"'] \arrow[dl, "b'"] & Y & \\ & & & \\ S' \arrow[rrr, "a'"] & & & T' \arrow[uul, "c'"'] \end{tikzcd}.$$

Note that when $\mathcal{C}$ is strict, $\square\mathcal{C}$ will also be strict.

Because $\mathcal{C}$ has biproducts, each object in $\mathcal{C}$ is a bicommutative bimonoid, and morphisms in $\mathcal{C}$ are all bimonoid homomorphisms. This definition of $\square\mathcal{C}$ generalizes the construction we used to form the PROP $\square(\texttt{FinVect}_k)$ from $\texttt{FinVect}_k$, but it still does not allow us to define $\square(\texttt{FinRel}_k)$. For that we need to drop the condition that $\mathcal{C}$ itself has biproducts, but we still require some of the structure that came with having biproducts. The rough idea is to find a subcategory $\mathcal{C}'$ of $\mathcal{C}$ that has all the objects of $\mathcal{C}$ such that $\mathcal{C}'$ has biproducts, define $\square\mathcal{C}'$, and bootstrap up to $\square\mathcal{C}$. Generally, a subcategory 'having all the objects' of a given category is not preserved by equivalences of categories, but there is a good alternative.

Definition 39 *An* **essentially wide** *subcategory $\mathcal{C}'$ of $\mathcal{C}$ is a subcategory of $\mathcal{C}$ that 'essentially' contains all the objects of $\mathcal{C}$. That is, the inclusion functor from $\mathcal{C}'$ to $\mathcal{C}$ is essentially surjective on objects.*

In PROPs and other strict categories, where isomorphic objects are equal, the notion of an essentially wide subcategory can be replaced with the notion of a *wide* subcategory, also referred to as a *lluf* subcategory. In this case 'essentially surjective' in the above definition is replaced with 'bijective'. That is, every object in $\mathcal{C}$ is also an object in any wide subcategory of $\mathcal{C}$.

Definition 40 *Given a category $(\mathcal{C}, m, 0, \Delta, !)$ with an (essentially) wide subcategory $(\mathcal{C}', m, 0, \Delta, !)$ such that $\mathcal{C}'$ has biproducts, we take $\Box\mathcal{C}$ to have*

- *the same objects as $\mathcal{C}$,*
- *the morphisms in $\hom(X, Y)$ as equivalence classes of*

$$X \xrightarrow{\Delta} X \otimes X \xrightarrow{\mathrm{id}_X \otimes b} X \otimes S \xrightarrow{\mathrm{id}_X \otimes a} X \otimes T \xrightarrow{d \otimes c} Y \otimes Y \xrightarrow{m} Y\ ,$$

  abbreviated as (d, c, a, b), such that $\Delta \circ c = \begin{bmatrix} c \\ c \end{bmatrix} \circ \Delta$, $\Delta \circ d = \begin{bmatrix} d \\ d \end{bmatrix} \circ \Delta$, $b \circ m = m \circ [b \quad b]$, and $d \circ m = m \circ [d \quad d]$,

- *composition given by*

$$(d, c, a, b) \circ (d', c', a', b') = \left(d'd, [d'c \quad c'], \begin{bmatrix} a & 0 \\ a'b'ca & a' \end{bmatrix}, \begin{bmatrix} b \\ b'd \end{bmatrix} \right).$$

Again, the morphisms of $\Box\mathcal{C}$ can be depicted as non-commuting squares:

$$\begin{tikzcd}
S \arrow[r, "a"] & T \arrow[d, "c"] \\
X \arrow[u, "b"] \arrow[r, "d"'] & Y
\end{tikzcd}.$$

The technical conditions on b, c, and d introduced in this version of $\square\mathcal{C}$ can be summarized as b is a monoid homomorphism, c is a comonoid homomorphism, and d is a bimonoid homomorphism. These will be discussed further at the end of this section, together with a depiction of the morphisms in $\square\mathcal{C}$ as string diagrams. The equivalence classes for morphisms in $\square\mathcal{C}$ are similar to what they were above. Two squares, (d, c, a, b) and (d, c', a', b') are in the same equivalence class if there are isomorphisms $\alpha\colon S \to S'$ and $\omega\colon T \to T'$ such that the following diagram in $\mathcal{C}$ commutes:

$$\begin{tikzcd}
S \arrow[rrr, "a"] \arrow[dd, "\alpha"'] & & & T \arrow[dl, "c"] \arrow[dd, "\omega"] \\
& X \arrow[ul, "b"] \arrow[dl, "b'"] & Y & \\
S' \arrow[rrr, "a'"] & & & T' \arrow[ul, "c'"]
\end{tikzcd}.$$

In what follows, the term $\square\mathcal{C}$ will refer to this more general version of $\square\mathcal{C}$ unless otherwise noted.

We refer to the objects S and T as the *prestate* and *state*, respectively. This should recall their names when these objects were vector spaces. In that case we referred to them as *prestate space* and *state space*, respectively. The formula for composition in $\square\mathcal{C}$ is easier to understand as coming from the diagram:

$$
\begin{tikzcd}
 & & S_1 \otimes S_2 \arrow[rr, "a \otimes a'"] & & T_1 \otimes T_2 \arrow[dll, "\pi_1"] \arrow[drr, "\pi_2"] & & \\
S_1 \arrow[urr, "\iota_1"] \arrow[rr, "a"] & & T_1 \arrow[dr, "c"'] & & S_2 \arrow[ull, "\iota_2"'] \arrow[rr, "a'"] & & T_2 \arrow[dl, "c'"] \\
 & X_1 \arrow[ul, "b"] \arrow[rr, "d"'] & & X_2 \arrow[ur, "b'"'] \arrow[rr, "d'"'] & & X_3 &
\end{tikzcd}
.
$$

The b and c sides in the composite come from the pentagons on the left and right, respectively. That is, b comes from the ways to get from X_1 to $S_1 \otimes S_2$, and c comes from the ways $T_1 \otimes T_2$ can get to X_3. Similarly, the a side in the composite comes from the ways $S_1 \otimes S_2$ can get to $T_1 \otimes T_2$, which includes a direct path (from S_1 to T_1 and from S_2 to T_2) and a looped path (from S_1 to T_2). The d side in the composite comes from the most direct path from X_1 to X_3.

Given two composable morphisms, $f\colon X_1 \to X_2$ and $g\colon X_2 \to X_3$ with representatives (d_1, c_1, a_1, b_1) and (d_1', c_1', a_1', b_1') for f and (d_2, c_2, a_2, b_2) and (d_2', c_2', a_2', b_2') for g, we have the following commuting diagrams:

$$
\begin{tikzcd}
S_1 \arrow[rrr, "a_1"] \arrow[dd, "\alpha_1"'] & & & T_1 \arrow[dl, "c_1"'] \arrow[dd, "\omega_1"] \\
 & X_1 \arrow[ul, "b_1"'] \arrow[dl, "b_1'"] & X_2 & \\
S_1' \arrow[rrr, "a_1'"] & & & T_1' \arrow[ul, "c_1'"']
\end{tikzcd}
$$

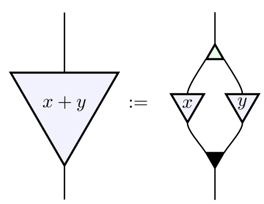
and

$$\begin{tikzcd}
S_2 \arrow[rrr, "a_2"] \arrow[dd, "\alpha_2"'] & & & T_2 \arrow[dl, "c_2"'] \arrow[dd, "\omega_2"] \\
& X_2 \arrow[ul, "b_2"'] \arrow[dl, "b_2'"] & X_3 & \\
S_2' \arrow[rrr, "a_2'"] & & & T_2' \arrow[ul, "c_2'"']
\end{tikzcd}.$$

We need isomorphisms $\alpha_{12} \colon S_1 \otimes S_2 \to S_1' \otimes S_2'$ and $\omega_{12} \colon T_1 \otimes T_2 \to T_1' \otimes T_2'$ that make the corresponding diagram for $g \circ f$ commute. We leave it as an exercise to the reader to check that $\alpha_{12} = \alpha_1 \otimes \alpha_2$ and $\omega_{12} = \omega_1 \otimes \omega_2$.

For the matrix notation of composition to make sense, there need to be notions of addition and multiplication. Multiplication is simply composition in $\mathcal{C}$. Addition is a generalization of equation **(12)** from Section 3.1, given in Definition 41.

Definition 41 *Let $(\mathcal{C}, m, 0, \Delta, !)$ be a category with an (essentially) wide subcategory $(\mathcal{C}', m, 0, \Delta, !)$ such that $\mathcal{C}'$ has biproducts, and $A, B \in \mathrm{Ob}(\mathcal{C})$. For $x, y \colon A \to B$ we define the operation $x + y := m_B \circ (x \otimes y) \circ \Delta_A$*



.

It is clear that if x, y are endomorphisms of A, $x + y$ will also be an endomorphism of A.

Theorem 42 *$\Box\mathcal{C}$ is a monoidal category with the same monoidal product on objects as $\mathcal{C}$, and $(d, c, a, b) \otimes (d', c', a', b') = (d \otimes d', c \otimes c', a \otimes a', b \otimes b')$.*

Proof. To show $\Box\mathcal{C}$ is a category, we need to show associativity and the unit laws hold. To show $\Box\mathcal{C}$ is a monoidal category, we also need to show the associator and unitors exist and satisfy the pentagon and triangle equations. It is clear that identity morphisms are formed when the prestate and state are both the zero object and d is an identity morphism in $\mathcal{C}$:

$$
\begin{tikzcd}
0 \arrow[r] & 0 \arrow[d] \\
X \arrow[u] \arrow[r, "1_X"'] & X
\end{tikzcd}
.
$$

The associator and unitors of $\square\mathcal{C}$ are formed from those of $\mathcal{C}$ using the same trick. It's easy to see their pentagon and triangle equations follow directly from those in $\mathcal{C}$. It is also easy to see the monoidal product on morphisms is compatible with composition. Associativity of morphisms in $\square\mathcal{C}$ reduces to associativity of the monoidal product for the prestate and state, and an associativity requirement on the $\mathcal{C}$ morphisms d, c, a, and b.

Denoting the compositions $(d_j, c_j, a_j, b_j) \circ (d_i, c_i, a_i, b_i)$ as $(d_{ij}, c_{ij}, a_{ij}, b_{ij})$, we get $(d_3, c_3, a_3, b_3) \circ (d_{12}, c_{12}, a_{12}, b_{12}) = (d_{12,3}, c_{12,3}, a_{12,3}, b_{12,3})$:

$$
\begin{tikzcd}
S_1 \otimes S_2 \arrow[rr, "a_{12}"] & & T_1 \otimes T_2 \arrow[dr, "c_{12}"'] & & S_3 \arrow[rr, "a_3"] & & T_3 \arrow[dl, "c_3"] \\
& X_1 \arrow[ul, "b_{12}"] \arrow[rr, "d_{12}"'] & & X_3 \arrow[ur, "b_3"'] \arrow[rr, "d_3"'] & & X_4
\end{tikzcd}
=
\begin{tikzcd}
(S_1 \otimes S_2) \otimes S_3 \arrow[r, "a_{12,3}"] & (T_1 \otimes T_2) \otimes T_3 \arrow[d, "c_{12,3}"] \\
X_1 \arrow[u, "b_{12,3}"] \arrow[r, "d_{12,3}"'] & X_4
\end{tikzcd}
$$

and $(d_{23}, c_{23}, a_{23}, b_{23}) \circ (d_1, c_1, a_1, b_1) = (d_{1,23}, c_{1,23}, a_{1,23}, b_{1,23})$:

$$
\begin{tikzcd}
S_1 \arrow[rr, "a_1"] & & T_1 \arrow[dr, "c_1"'] & & S_2 \otimes S_3 \arrow[rr, "a_{23}"] & & T_2 \otimes T_3 \arrow[dl, "c_{23}"] \\
& X_1 \arrow[ul, "b_1"] \arrow[rr, "d_1"'] & & X_2 \arrow[ur, "b_{23}"'] \arrow[rr, "d_{23}"'] & & X_4
\end{tikzcd}
=
\begin{tikzcd}
S_1 \otimes (S_2 \otimes S_3) \arrow[r, "a_{1,23}"] & T_1 \otimes (T_2 \otimes T_3) \arrow[d, "c_{1,23}"] \\
X_1 \arrow[u, "b_{1,23}"] \arrow[r, "d_{1,23}"'] & X_4
\end{tikzcd}
.
$$

The associativity requirements for d, c, a, and b require there to be canonical isomorphisms $d_{12,3} \cong d_{1,23}$, $c_{12,3} \cong c_{1,23}$, $a_{12,3} \cong a_{1,23}$, and $b_{12,3} \cong b_{1,23}$. The associativity requirement for d clearly holds because composition is associative in $\mathcal{C}$ — $d_{1,23} = d_1(d_2 d_3) \cong (d_1 d_2) d_3 = d_{12,3}$. We see the associativity requirement for a holds because $a_{ij} = \begin{bmatrix} a_i & 0 \\ a_j b_j c_i a_i & a_j \end{bmatrix}$, which means

$$a_{12,3} = \begin{bmatrix} a_{12} & 0 \\ a_3b_3c_{12}a_{12} & a_3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_1 & 0 \\ a_2b_2c_1a_1 & a_2 \end{bmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ [a_3b_3d_2c_1a_1 + a_3b_3c_2a_2b_2c_1a_1 \quad a_3b_3c_2a_2] & a_3 \end{bmatrix},$$

since $c_{12} = \begin{bmatrix} d_2c_1 & c_2 \end{bmatrix}$. A similar calculation gives a canonically isomorphic matrix for $a_{1,23}$,

$$a_{1,23} = \begin{bmatrix} a_1 & \begin{matrix} 0 & 0 \end{matrix} \\ \begin{bmatrix} a_2b_2c_1a_1 \\ a_3b_3d_2c_1a_1 + a_3b_3c_2a_2b_2c_1a_1 \end{bmatrix} & \begin{bmatrix} a_2 & 0 \\ a_3b_3c_2a_2 & a_3 \end{bmatrix} \end{bmatrix}.$$

The proofs of the associativity requirements for c and b are similar to each other, transposed. We present the argument for c and leave the argument for b to the reader. Since $c_{ij} = [d_jc_i \quad c_j]$, we have

$$\begin{aligned} c_{12,3} &= [d_3c_{12} \quad c_3] \\ &= [d_3[d_2c_1 \quad c_2] \quad c_3] \\ &= [[d_3d_2c_1 \quad d_3c_2] \quad c_3] \\ &\cong [d_3d_2c_1 \quad [d_3c_2 \quad c_3]] \\ &= [d_{23}c_1 \quad c_{23}] = c_{1,23}. \end{aligned}$$

So we see composition of morphisms in $\Box\mathcal{C}$ is associative. ■

The anatomy of $\Box\mathcal{C}$ makes more sense when it is understood as a category with an obvious evaluation functor $\mathrm{eval}\colon \Box\mathcal{C} \to \mathcal{C}$. We can also find a 'feedthrough' functor $\mathrm{feed}\colon \Box\mathcal{C} \to \mathcal{C}$ and functor in the reverse direction $\mathrm{Box}\colon \mathcal{C} \to \Box\mathcal{C}$. The maps of objects $\mathrm{eval}_0\colon \mathrm{Ob}(\Box\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$ is a bijection, $\mathrm{feed}_0 = \mathrm{eval}_0$, and Box_0 is its inverse. The map of morphisms $\mathrm{eval}_1\colon \mathrm{Mor}(\Box\mathcal{C}) \to \mathrm{Mor}(\mathcal{C})$ is given by $\mathrm{eval}_1(d,c,a,b) = d + cab$, $\mathrm{feed}_1\colon \mathrm{Mor}(\Box\mathcal{C}) \to \mathrm{Mor}(\mathcal{C})$ is given by $\mathrm{feed}_1(d,c,a,b) = d$, and $\mathrm{Box}_1\colon \mathrm{Mor}(\mathcal{C}) \to \mathrm{Mor}(\Box\mathcal{C})$ is given by $\mathrm{Box}_1(d) = (d,!,0,0)$. That is,

$$\mathrm{eval}_1\left(\begin{array}{ccc} S & \xrightarrow{a} & T \\ {\scriptstyle b}\uparrow & & \downarrow{\scriptstyle c} \\ X_1 & \xrightarrow[d]{} & X_2 \end{array}\right) = d + cab, \quad \mathrm{feed}_1\left(\begin{array}{ccc} S & \xrightarrow{a} & T \\ {\scriptstyle b}\uparrow & & \downarrow{\scriptstyle c} \\ X_1 & \xrightarrow[d]{} & X_2 \end{array}\right) = d$$

$$\text{and } \mathrm{Box}_1(d) = \begin{array}{ccc} 0 & \longrightarrow & 0 \\ \uparrow & & \downarrow \\ X_1 & \xrightarrow[d]{} & X_2 \end{array}.$$

Theorem 43 eval*,* feed *and* Box *are monoidal functors, and*

- eval*,* feed *and* Box *are essentially surjective*
- eval *and* feed *are full, but not faithful*
- Box *is faithful, but not full*
- Box *has no adjoint.*
- *If $\mathcal{C}$ is a symmetric monoidal category (*resp. *a braided monoidal category),* eval*,* feed *and* Box *are symmetric (*resp. *braided) monoidal functors. In particular, $\Box\mathcal{C}$ will also be a symmetric (*resp. *braided) monoidal category.*

Note that since $\Box\mathcal{C}$ is strict whenever $\mathcal{C}$ is, this last item means $\Box\mathcal{C}$ is a PROP whenever $\mathcal{C}$ is.

Proof. All three functors are bijective on objects, so it immediately follows they are essentially surjective. We note that feed$\circ$Box and eval$\circ$Box are both the identity functor on $\mathcal{C}$, which implies feed and eval are surjective on *all* morphisms, hence full. This also implies Box is injective on morphisms, so Box is faithful. On the other hand, a morphism in $\Box\mathcal{C}$ between $\mathrm{Box}_0(V_1)$ and $\mathrm{Box}_0(V_2)$ where the prestate or state are not isomorphic to the zero object is not the Box_1-image of any morphism in $\mathcal{C}$, so Box is not full. Similarly, feed and eval cannot be faithful.

$$
\begin{tikzcd}
 & V_1 \otimes (V_2 \otimes V_3) \arrow[r, "{(B_{V_1, V_2 \otimes V_3}, !, 0, 0)}"] & (V_2 \otimes V_3) \otimes V_1 \arrow[dr, "{(a_{V_2,V_3,V_1}, !, 0, 0)}"] & \\
(V_1 \otimes V_2) \otimes V_3 \arrow[ur, "{(a_{V_1,V_2,V_3}, !, 0, 0)}"] \arrow[dr, "{(B_{V_1,V_2} \otimes \mathrm{Id}, !, 0, 0)}"'] & & & V_2 \otimes (V_3 \otimes V_1) \\
 & (V_2 \otimes V_1) \otimes V_3 \arrow[r, "{(a_{V_2,V_1,V_3}, !, 0, 0)}"'] & V_2 \otimes (V_1 \otimes V_3) \arrow[ur, "{(\mathrm{Id} \otimes B_{V_1,V_3}, !, 0, 0)}"'] &
\end{tikzcd}
$$

Figure B.1: A hexagon law inside $\Box\mathcal{C}$. This diagram commutes in $\Box\mathcal{C}$ when the analogous diagram in $\mathcal{C}$, where the morphisms are the first coordinates of the morphisms here, commutes.

In Theorem 18 we saw Box has no adjoint, taking advantage of the fact that $\texttt{FinVect}_k$ has an initial object and a terminal object, neither of which is preserved by Box. In this more general setting, $\mathcal{C}$ may not have initial or terminal objects, so it is necessary to show from the definitions that Box has no adjoint. It is a straightforward exercise sketched out below, which we leave to the reader to fill in the details.

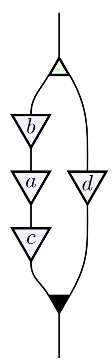
Suppose $R\colon \Box\mathcal{C} \to \mathcal{C}$, $d \in \hom_{\mathcal{C}}(A, B)$, and $R(d, c, a, b) = f \in \hom_{\mathcal{C}}(A, B)$. Clearly $\mathrm{Box} \circ R(d, c, a, b) = (f, !, 0, 0)$. Further suppose $a \in \hom_{\mathcal{C}}(P, S)$, so that the prestate of (d, c, a, b) is P and the state is S. If R is a right adjoint to Box, there would be a natural transformation $\eta\colon 1_{\Box\mathcal{C}} \Rightarrow \mathrm{Box} \circ R$ such that the square

$$
\begin{tikzcd}
A \arrow[r, "{(d, c, a, b)}"] \arrow[d, "\eta_A"'] & B \arrow[d, "\eta_B"] \\
A \arrow[r, "{(f, !, 0, 0)}"'] & B
\end{tikzcd}
$$

commutes in $\Box\mathcal{C}$. Taking P' and P'' to be the prestates for η_A and η_B, respectively, this means $P' \cong P'' \oplus P$. However, P can vary without affecting A or B, while P' and P'' are determined by A and B. This contradiction means R cannot be a right adjoint to Box. Similarly Box has no left adjoint, by considering the natural transformation $\epsilon\colon \mathrm{Box} \circ L \Rightarrow 1_{\square\mathcal{C}}$.

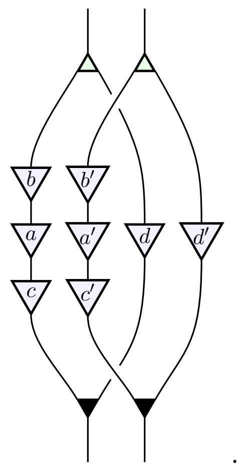
It is easy to check that the associator, unitor, and symmetry/braiding isomorphisms in $\square\mathcal{C}$ are the Box_1-images of the respective isomorphisms in $\mathcal{C}$. In $\square\mathcal{C}$ we have $(d, !, 0, 0) \circ (d', !, 0, 0) = (d \circ d', !, 0, 0)$, so Box preserves the coherence laws that hold in $\mathcal{C}$. See Figure B.1 for an example of one of these coherence laws in $\square\mathcal{C}$. Thus $\square\mathcal{C}$ is a symmetric (*resp.* braided) monoidal category when $\mathcal{C}$ is. It is also easy to see $\mathrm{feed}_1(d, !, 0, 0) = \mathrm{eval}_1(d, !, 0, 0) = d$, so feed and eval will also preserve all the coherence laws. ■

An alternate way to depict morphisms in $\square\mathcal{C}$ is through string diagrams. The morphism (d, c, a, b) is depicted:

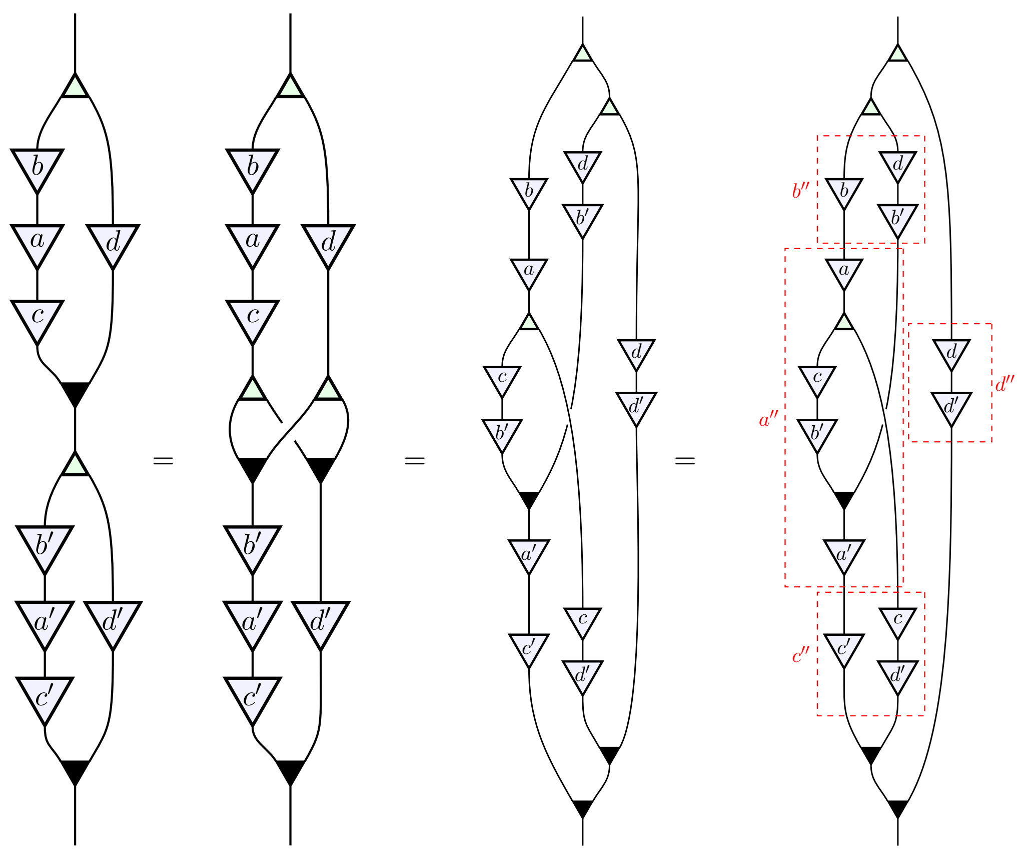


,

and it is easy to see the monoidal product $(d, c, a, b) \otimes (d', c', a', b')$ is what it is supposed to be:



Composition is still tedious with string diagrams, but the reason for the technical conditions on b, c, and d is illuminated somewhat:



The first equality is because every object is a bimonoid. The second equality is due to the technical conditions and topology-preserving moves. The last equality is due to every object being bicommutative. The dashed boxes indicate which portions of the string diagram correspond to the components of the composite morphism: $(d', c', a', b') \circ (d, c, a, b) = (d'', c'', a'', b'')$.