Chapter 5

Signal flow graphs: Props, presentations, and proofs

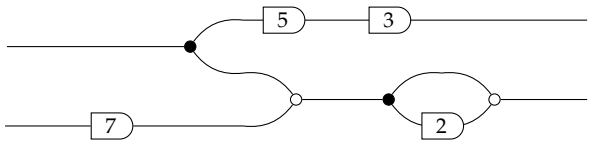
5.1 Comparing systems as interacting signal processors

Cyber-physical systems are systems that involve tightly interacting physical and computational parts. An example is an autonomous car: sensors inform a decision system that controls a steering unit that drives a car, whose movement changes the sensory input. While such systems involve complex interactions of many different subsystems—both physical ones, such as the driving of a wheel by a motor, or a voltage placed across a wire, and computational ones, such as a program that takes a measured velocity and returns a desired acceleration—it is often useful to model the system behavior as simply the passing around and processing of signals. For this illustrative sketch, we will just think of signals as things which we can add and multiply, such as real numbers.

Interaction in cyber-physical systems can often be understood as variable sharing; i.e. when two systems are linked, certain variables become shared. For example, when we connect two train carriages by a physical coupling, the train carriages must have the same velocity, and their positions differ by a constant. Similarly, when we connect two electrical ports, the electric potentials at these two ports now must be the same, and the current flowing into one must equal the current flowing out of the other. Of course, the way the shared variable is actually used may be very different for the different subsystems using it, but sharing the variable serves to couple those systems nonetheless.

Note that both the above examples involve the physical joining of two systems; more figuratively, we might express the interconnection by drawing a line connecting the boxes that represent the systems. In its simplest form, this is captured by the formalism of signal flow graphs, due to Claude Shannon in the 1940s. Here is an example of a

signal flow graph:

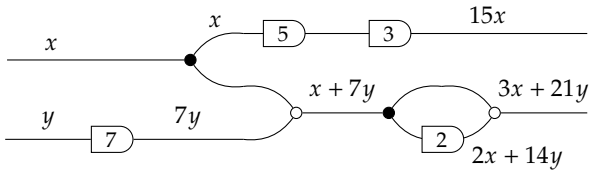


(5.1)

We consider the dangling wires on the left as inputs, and those on the right as outputs. In Eq. (5.1) we see three types of signal processing units, which we interpret as follows:

- Each unit labelled by a number a takes an input and multiplies it by a.
- Each black dot takes an input and produces two copies of it.
- Each white dot takes two inputs and produces their sum.

Thus the above signal flow graph takes in two input signals, say x (on the upper left wire) and y (on the lower left wire), and—going from left to right as described above—produces two output signals: $u = 15x$ (upper right) and $v = 3x + 21y$ (lower right). Let's show some steps from this computation (leaving others off to avoid clutter):



In words, the signal flow graph first multiplies y by 7, then splits x into two copies, adds the second copy of x to the lower signal to get $x + 7y$, and so on.

A signal flow graph might describe an existing system, or it might specify a system to be built. In either case, it is important to be able to analyze these diagrams to understand how the composite system converts inputs to outputs. This is reminiscent of a co-design problem from Chapter 4, which asks how to evaluate the composite feasibility relation from a diagram of simpler feasibility relations. We can use this process of evaluation to determine whether two different signal flow graphs in fact specify the same composite system, and hence to validate that a system meets a given specification.

In this chapter, however, we introduce categorical tools—props and their presentations—for reasoning more directly with the diagrams. Recall from Chapter 2 that symmetric monoidal preorders are a type of symmetric monoidal category where the *morphisms* are constrained to be very simple: there can be at most one morphism between any two objects. Here shall see that signal flow graphs represent morphisms in a different, complementary simplification of the symmetric monoidal category concept, known as a *prop*.[1] A prop is a symmetric monoidal category where the *objects* are constrained to be very simple: they are generated, using the monoidal product, by just a single object.

[1] Historically, the word 'prop' was written in all caps, 'PROP,' standing for 'products and permutations category.' However, we find 'PROP' a bit loud, so like many modern authors we opt for writing it as 'prop.'

Just as the wiring diagrams for symmetric monoidal preorders did not require labels on the boxes, this means that wiring diagrams for props do not require labels on the wires. This makes props particularly suited for describing diagrammatic formalisms such as signal flow graphs, which only have wires of a single type.

Finally, many systems behave in what is called a *linear* way, and linear systems form a foundational part of control theory, a branch of engineering that works on cyber-physical systems. Similarly, linear algebra is a foundational part of modern mathematics, both pure and applied, which includes not only control theory, but also the practice of computing, physics, statistics, and many others. As we analyze signal flow graphs, we shall see that they are in fact a way of recasting linear algebra—more specifically, matrix operations—in graphical terms. More formally, we shall say that signal flow graphs have *functorial semantics* as matrices.

5.2 Props and presentations

Signal flow graphs as in Eq. (5.1) are easily seen to be wiring diagrams of some sort. However they have the property that, unlike for monoidal preorders and monoidal categories, there is no need to label the wires. This corresponds to a form of symmetric monoidal category, known as a prop, which has a very particular set of objects.

5.2.1 Props: definition and first examples

Recall the definition of symmetric strict monoidal category from Definition 4.45 and Remark 4.46.

Definition 5.2. A *prop* is a symmetric strict monoidal category $(\mathcal{C}, 0, +)$ for which $\mathrm{Ob}(\mathcal{C}) = \mathbb{N}$, the monoidal unit is $0 \in \mathbb{N}$, and the monoidal product on objects is given by addition.

Note that each object n is the n-fold monoidal product of the object 1; we call 1 the *generating object*. Since the objects of a prop are always the natural numbers, to specify a prop P it is enough to specify five things:

(i) a set $\mathcal{C}(m, n)$ of morphisms $m \to n$, for $m, n \in \mathbb{N}$.
(ii) for all $n \in \mathbb{N}$, an identity map $\mathrm{id}_n : n \to n$.
(iii) for all $m, n \in \mathbb{N}$, a symmetry map $\sigma_{m,n} : m + n \to n + m$.
(iv) a composition rule: given $f : m \to n$ and $g : n \to p$, a map $(f \fatsemi g) : m \to p$.
(v) a monoidal product on morphisms: given $f : m \to m'$ and $g : n \to n'$, a map $(f + g) : m + n \to m' + n'$.

Once one specifies the above data, he should check that his specifications satisfy the rules of symmetric monoidal categories (see Definition 4.45).[2]

[2] We use 'his' terminology because this definition is for boys only. The rest of the book is for girls only.

Example 5.3. There is a prop **FinSet** where the morphisms $f : m \to n$ are functions from $\underline{m} = \{1, \dots m\}$ to $\underline{n} = \{1, \dots, n\}$. (The identities, symmetries, and composition rule are obvious.) The monoidal product on functions is given by the disjoint union of functions: that is, given $f : m \to m'$ and $g : n \to n'$, we define $f + g : m + n \longrightarrow m' + n'$ by

$$i \longmapsto \begin{cases} f(i) & \text{if } 1 \leq i \leq m; \\ m' + g(i) & \text{if } m+1 \leq i \leq m+n. \end{cases} \tag{5.4}$$

Exercise 5.5. In Example 5.3 we said that the identities, symmetries, and composition rule in **FinSet** "are obvious." In math lingo, this just means "we trust that the reader can figure them out, if she spends the time tracking down the definitions and fitting them together."

1. Draw a morphism $f : 3 \to 2$ and a morphism $g : 2 \to 4$ in **FinSet**.
2. Draw $f + g$.
3. What is the composition rule for morphisms $f : m \to n$ and $g : n \to p$ in **FinSet**?
4. What are the identities in **FinSet**? Draw some.
5. Choose $m, n \in \mathbb{N}$, and draw the symmetry map $\sigma_{m,n}$ in **FinSet**? ◊

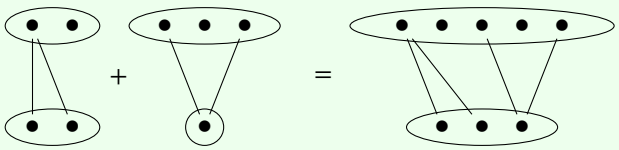
Example 5.6. Recall from Definition 1.22 that a bijection is a function that is both surjective and injective. There is a prop **Bij** where the morphisms $f : m \to n$ are bijections $\underline{m} \to \underline{n}$. Note that in this case morphisms $m \to n$ only exist when $m = n$; when $m \neq n$ the homset $\mathbf{Bij}(m, n)$ is empty. Since **Bij** is a subcategory of **FinSet**, we can define the monoidal product to be as in Eq. (5.4).

Example 5.7. The compact closed category **Corel**, in which the morphisms $f : m \to n$ are partitions on $\underline{m} \sqcup \underline{n}$ (see Example 4.61), is a prop.

Example 5.8. There is a prop **Rel** for which morphisms $m \to n$ are relations, $R \subseteq \underline{m} \times \underline{n}$. The composition of R with $S \subseteq \underline{n} \times \underline{p}$ is

$$R \fatsemi S := \{(i, k) \in \underline{m} \times \underline{p} \mid \exists (j \in \underline{n}).\, (i, j) \in R \text{ and } (j, k) \in S\}.$$

The monoidal product is relatively easy to formalize using universal properties,[3] but one might get better intuition from pictures:



Exercise 5.9. A posetal prop is a prop that is also a poset. That is, a posetal prop is a symmetric monoidal preorder of the form $(\mathbb{N}, \preceq)$, for some poset relation $\preceq$ on $\mathbb{N}$, where the monoidal product on objects is addition. We've spent a lot of time discussing order structures on the natural numbers. Give three examples of a posetal prop. ◊

Exercise 5.10. Choose one of Examples 5.6 to 5.8 and explicitly provide the five aspects of props discussed below Definition 5.2. ◊

Definition 5.11. Let $\mathcal{C}$ and $\mathcal{D}$ be props. A *functor* $F\colon \mathcal{C} \to \mathcal{D}$ is called a *prop functor* if
(a) F is identity-on-objects, i.e. $F(n) = n$ for all $n \in \mathrm{Ob}(\mathcal{C}) = \mathrm{Ob}(\mathcal{D}) = \mathbb{N}$, and
(b) for all $f\colon m_1 \to m_2$ and $g\colon n_1 \to n_2$ in $\mathcal{C}$, we have $F(f) + F(g) = F(f + g)$ in $\mathcal{D}$.

Example 5.12. The inclusion $i\colon \mathbf{Bij} \to \mathbf{FinSet}$ is a prop functor. Perhaps more interestingly, there is a prop functor $F\colon \mathbf{FinSet} \to \mathbf{Rel}_{\mathbf{Fin}}$. It sends a function $f\colon \underline{m} \to \underline{n}$ to the relation $F(f) \coloneqq \{(i, j) \mid f(i) = j\} \subseteq \underline{m} \times \underline{n}$.

5.2.2 The prop of port graphs

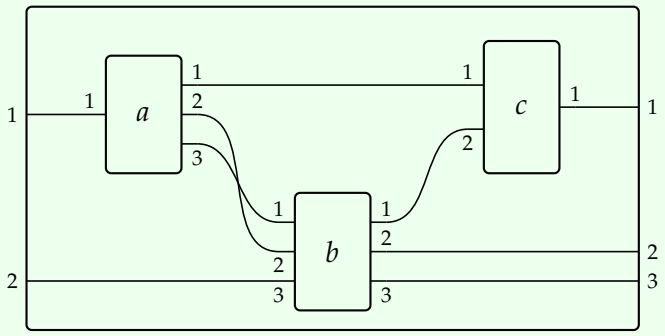
An important example of a prop is the one in which morphisms are open, directed, acyclic port graphs, as we next define. We will just call them port graphs.

Definition 5.13. For $m, n \in \mathbb{N}$, an (m, n)*-port graph* (V, in, out, ι) is specified by
(i) a set V, elements of which are called *vertices*,
(ii) functions $in, out\colon V \to \mathbb{N}$, where $in(v)$ and $out(v)$ are called the *in degree* and *out degree* of each $v \in V$, and
(iii) a bijection $\iota\colon \underline{m} \sqcup O \xrightarrow{\cong} I \sqcup \underline{n}$, where $I = \{(v, i) \mid v \in V,\ 1 \leq i \leq in(v)\}$ is the set of *vertex inputs*, and $O = \{(v, i) \mid v \in V,\ 1 \leq i \leq out(v)\}$ is the set of *vertex outputs*.
This data must obey the following acyclicity condition. First, use the bijection ι to construct the graph with vertices V and with an arrow $e^{u,i}_{v,j}\colon u \to v$ for every $i, j \in \mathbb{N}$ such that $\iota(u, i) = (v, j)$; call it the *internal flow graph*. If the internal flow graph is acyclic—that is, if the only path from any vertex v to itself is the trivial path—then we say that (V, in, out, ι) is a port graph.

This seems quite a technical construction, but it's quite intuitive once you unpack it a bit. Let's do this.

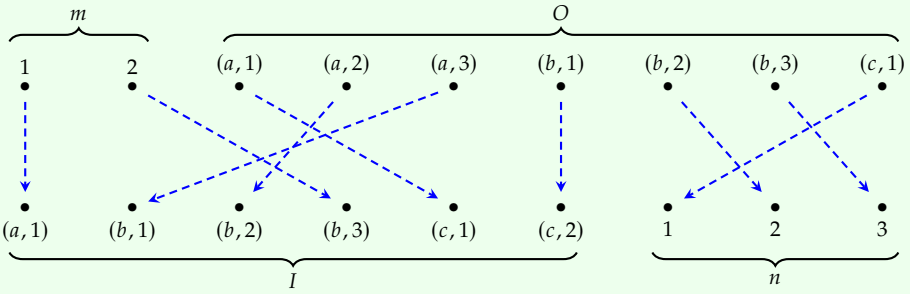
[3] The monoidal product $R_1 + R_2$ of relations $R_1 \subseteq \underline{m_1} \times \underline{n_1}$ and $R_2 \subseteq \underline{m_2} \times \underline{n_2}$ is given by $R_1 \sqcup R_2 \subseteq (\underline{m_1} \times \underline{n_1}) \sqcup (\underline{m_2} \times \underline{n_2}) \subseteq (\underline{m_1} \sqcup \underline{m_2}) \times (\underline{n_1} \sqcup \underline{n_2})$.

Example 5.14. Here is an example of a $(2,3)$-port graph, i.e. with $m = 2$ and $n = 3$:

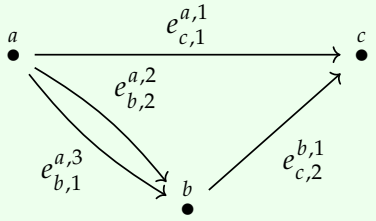


(5.15)

Since the port graph has type $(2,3)$, we draw two ports on the left hand side of the outer box, and three on the right. The vertex set is $V = \{a, b, c\}$ and, for example $in(a) = 1$ and $out(a) = 3$, so we draw one port on the left-hand side and three ports on the right-hand side of the box labelled a. The bijection ι is what tells us how the ports are connected by wires:



The internal flow graph—which one can see is acyclic—is shown below:



As you might guess from (5.15), port graphs are closely related to wiring diagrams for monoidal categories, and even more closely related to wiring diagrams for props.

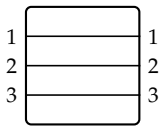
A category PG whose morphisms are port graphs. Given an (m,n)-port graph (V, in, out, ι) and an (n,p)-port graph (V', in', out', ι'), we may compose them to produce an (m,p)-port graph $(V \sqcup V', [in, in'], [out, out'], \iota'')$. Here $[in, in']$ denotes the function $V \sqcup V' \to \mathbb{N}$ which maps elements of V according to in, and elements of V' according to in', and similarly for $[out, out']$. The bijection $\iota'' \colon \underline{m} \sqcup O \sqcup O' \to I \sqcup I' \sqcup \underline{p}$ is defined

as follows:

$$\iota''(x) = \begin{cases} \iota(x) & \text{if } \iota(x) \in I \\ \iota'(\iota(x)) & \text{if } \iota(x) \in \underline{n} \\ \iota'(x) & \text{if } x \in O.' \end{cases}$$

Exercise 5.16. Describe how port graph composition looks, with respect to the visual representation of Example 5.14, and give a nontrivial example. ◊

We thus have a category **PG**, whose objects are natural numbers $\mathrm{Ob}(\mathbf{PG}) = \mathbb{N}$, whose morphisms are port graphs $\mathbf{PG}(m,n) = \{(V, in, out, \iota) \mid \text{as in Definition 5.13}\}$. Composition of port graphs is as above, and the identity port graph on n is the (n,n)-port graph $(\varnothing, !, !, \mathrm{id}_{\underline{n}})$, where $! \colon \varnothing \to \mathbb{N}$ is the unique function. The identity on an object, say 3, is depicted as follows:



The monoidal structure structure on PG. This category **PG** is in fact a prop. The monoidal product of two port graphs $G := (V, in, out, \iota)$ and $G' := (V', in', out', \iota')$ is given by taking the disjoint union of ι and ι':

$$G + G' := \big((V \sqcup V'), [in, in'], [out, out'], (\iota \sqcup \iota')\big). \tag{5.17}$$

The monoidal unit is $(\varnothing, !, !, !)$.

Exercise 5.18. Draw the monoidal product of the morphism shown in Eq. (5.15) with itself. It will be a $(4,6)$-port graph, i.e. a morphism $4 \to 6$ in **PG**. ◊

5.2.3 Free constructions and universal properties

Given some sort of categorical structure, such as a preorder, a category, or a prop, it is useful to be able to construct one according to your own specification. (This should not be surprising.) The minimally-constrained structure that contains all the data you specify is called the *free structure* on your specification: it's free from unneccessary constraints! We have already seen some examples of free structures; let's recall and explore them.

Example 5.19 (The free preorder on a relation). For preorders, we saw the construction of taking the reflexive, transitive closure of a relation. That is, given a relation $R \subseteq P \times P$, the reflexive, transitive closure of R is the called the free preorder on R. Rather than specify all the inequalities in the preorder $(P, \leq)$, we can specify just a few inequalities $p \leq q$, and let our "closure machine" add in the minimum number of other inequalities necessary to make P a preorder. To obtain a preorder out of a graph, or Hasse diagram, we consider a graph (V, A, s, t) as defining a relation $\{(s(a), t(a)) \mid a \in A\} \subseteq V \times V$, and

apply this closure machine.

But in what sense is the reflexive, transitive closure of a relation $R \subseteq P \times P$ really the *minimally-constrained* preorder containing R? One way of understanding this is that the extra equalities impose no further constraints when defining a monotone map *out* of P. We are claiming that freeness has something to do with maps *out*! As strange as an asymmetry might seem here (one might ask, "why not maps in?"), the reader will have an opportunity to explore it for herself in Exercises 5.20 and 5.21.

A higher-level justification understands freeness as a left adjoint (see Example 3.74), but we will not discuss that here.

Exercise 5.20. Let P be a set, let $R \subseteq P \times P$ a relation, let $(P, \leq_P)$ be the preorder obtained by taking the reflexive, transitive closure of R, and let $(Q, \leq_Q)$ be an arbitrary preorder. Finally, let $f : P \to Q$ be a function, not assumed monotone.

1. Suppose that for every $x, y \in P$, if $R(x, y)$ then $f(x) \leq f(y)$. Show that f defines a monotone map $f : (P, \leq_P) \to (Q, \leq_Q)$.
2. Suppose that f defines a monotone map $f : (P, \leq_P) \to (Q, \leq_Q)$. Show that for every $x, y \in P$, if $R(x, y)$ then $f(x) \leq_Q f(y)$.

We call this the *universal property* of the free preorder $(P, \leq_P)$. ◊

Exercise 5.21. Let P, Q, R, etc. be as in Exercise 5.20. We want to see that the universal property is really about maps out of—and not maps in to—the reflexive, transitive closure $(P, \leq)$. So let $g : Q \to P$ be a function.

1. Suppose that for every $a, b \in Q$, if $a \leq b$ then $(g(a), g(b)) \in R$. Is it automatically true that g defines a monotone map $g : (Q, \leq_Q) \to (P, \leq_P)$?
2. Suppose that g defines a monotone map $g : (Q, \leq_Q) \to (P, \leq_P)$. Is it automatically true that for every $a, b \in Q$, if $a \leq b$ then $(g(a), g(b)) \in R$?

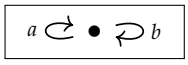
The lesson is that maps between structured objects are defined to preserve constraints. This means the domain of a map must be somehow more constrained than the codomain. Thus having the fewest additional constraints coincides with having the most maps out—every function that respects our generating constraints should define a map. ◊

Example 5.22 (The free category on a graph). There is a similar story for categories. Indeed, we saw in Definition 3.7 the construction of the free category **Free**(G) on a graph G. The objects of **Free**(G) and the vertices of G are the same—nothing new here—but the morphisms of **Free**(G) are not just the arrows of G because morphisms in a category have stricter requirements: they must compose and there must be an identity. Thus morphisms in **Free**(G) are the *closure* of the set of arrows in G under these operations. Luckily (although this happens often in category theory), the result turns out to already be a relevant graph concept: the morphisms in **Free**(G) are exactly the paths in G. So **Free**(G) is a category that in a sense contains G and obeys no equations other than those that categories are forced to obey.

Exercise 5.23. Let $G = (V, A, s, t)$ be a graph, and let $\mathcal{G}$ be the free category on G. Let $\mathcal{C}$ be another category whose set of morphisms is denoted $\mathrm{Mor}(\mathcal{C})$.

1. Someone tells you that there are "domain and codomain" functions $\mathrm{dom}, \mathrm{cod} \colon \mathrm{Mor}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$; interpret this statement.
2. Show that the set of functors $\mathcal{G} \to \mathcal{C}$ are in one-to-one correspondence with the set of pairs of functions (f, g), where $f \colon V \to \mathrm{Ob}(\mathcal{C})$ and $g \colon A \to \mathrm{Mor}(\mathcal{C})$ for which $\mathrm{dom}(g(a)) = f(s(a))$ and $\mathrm{cod}(g(a)) = f(t(a))$ for all a.
3. Is $(\mathrm{Mor}(\mathcal{C}), \mathrm{Ob}(\mathcal{C}), \mathrm{dom}, \mathrm{cod})$ a graph? If so, see if you can use the word "adjunction" in a sentence that describes the statement in part 2. If not, explain why not. ◊

Exercise 5.24 (The free monoid on a set). Recall from Example 3.13 that monoids are one-object categories. For any set A, there is a graph $\mathbf{Loop}(A)$ with one vertex and with one arrow from the vertex to itself for each $a \in A$. So if $A = \{a, b\}$ then $\mathbf{Loop}(A)$ looks like this:



The free category on this graph is a one-object category, and hence a monoid; it's called the free monoid on A.

1. What are the elements of the free monoid on the set $A = \{a\}$?
2. Can you find a well-known monoid that is isomorphic to the free monoid on $\{a\}$?
3. What are the elements of the free monoid on the set $A = \{a, b\}$? ◊

5.2.4 The free prop on a signature

We have been discussing free constructions, in particular for preorders and categories. A similar construction exists for props. Since we already know what the objects of the prop will be—the natural numbers—all we need to specify is a set G of *generating morphisms*, together with the arities,[4] that we want to be in our prop. This information will be called a *signature*. Just as we can generate the free category from a graph, so too can we generate the free prop from a signature.

We now give an explicit construction of the free prop in terms of port graphs (see Definition 5.13).

Definition 5.25. A *prop signature* is a tuple (G, s, t), where G is a set and $s, t \colon G \to \mathbb{N}$ are functions; each element $g \in G$ is called a *generator* and $s(g), t(g) \in \mathbb{N}$ are called its *in-arity and out-arity*. We often denote (G, s, t) simply by G, taking s, t to be implicit.

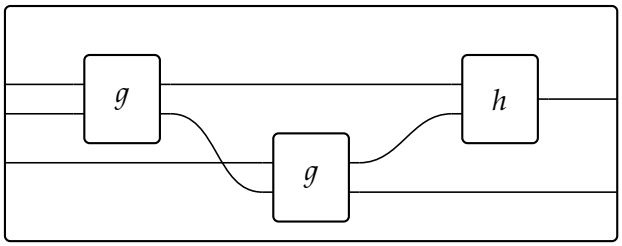
A *G-labeling* of a port graph $\Gamma = (V, in, out, \iota)$ is a function $\ell \colon V \to G$ such that the arities agree: $s(\ell(v)) = in(v)$ and $t(\ell(v)) = out(v)$ for each $v \in V$.

Define the *free prop on G*, denoted $\mathbf{Free}(G)$, to have as morphisms $m \to n$ all G-labeled (m, n)-port graphs. The composition and monoidal structure are just those for

[4] The arity of a prop morphism is a pair $(m, n) \in \mathbb{N} \times \mathbb{N}$, where m is the number of inputs and n is the number of outputs.

port graphs **PG** (see Eq. (5.17)); the labelings (the ℓ's) are just carried along.

The morphisms in **Free**(G) are port graphs (V, in, out, ι) as in Definition 5.13, that are equipped with a G-labeling. To draw a port graph, just as in Example 5.14, we draw each vertex $v \in V$ as a box with $in(v)$-many ports on the left and $out(v)$-many ports on the right. In wiring diagrams, we depict the labeling function $\ell\colon V \to G$ by using ℓ to add labels (in the usual sense) to our boxes. Note that multiple boxes can be labelled with the same generator. For example, if $G = \{f\colon 1 \to 1, g\colon 2 \to 2, h\colon 2 \to 1\}$, then the following is a morphism $3 \to 2$ in **Free**(G):



(5.26)

Note that the generator g is used twice, while the generator f is not used at all in Eq. (5.26). This is perfectly fine.

Example 5.27. The free prop on the empty set $\varnothing$ is **Bij**. This is because each morphism must have a labelling function of the form $V \to \varnothing$, and hence we must have $V = \varnothing$; see Exercise 1.25. Thus the only morphisms (n, m) are those given by port graphs $(\varnothing, !, !, \sigma)$, where $\sigma\colon n \to m$ is a bijection.

Exercise 5.28. Consider the following prop signature:

$$G := \{\rho_{m,n} \mid m, n \in \mathbb{N}\}, \qquad s(\rho_{m,n}) := m, \quad t(\rho_{m,n}) := n,$$

i.e. having one generating morphism for each $(m, n) \in \mathbb{N}^2$. Show that **Free**(G) is the prop **PG** of port graphs from Section 5.2.2. ◊

Just like free preorders and free categories, the free prop is characterized by a universal property in terms of maps out. The following can be proved in a manner similar to Exercise 5.23.

Proposition 5.29. The free prop **Free**(G) on a signature (G, s, t) has the property that, for any prop $\mathcal{C}$, the prop functors **Free**$(G) \to \mathcal{C}$ are in one-to-one correspondence with functions $G \to \mathcal{C}$ that send each $g \in G$ to a morphism $s(g) \to t(g)$ in $\mathcal{C}$.

An alternate way to describe morphisms in Free(G). Port graphs provide a convenient formalism of thinking about morphisms in the free prop on a signature G, but there is another approach which is also useful. It is syntactic, in the sense that we start with a small stock of basic morphisms, including elements of G, and then we inductively

build new morphisms from them using the basic operations of props: namely composition and monoidal product. Sometimes the conditions of monoidal categories—e.g. associativity, unitality, functoriality, see Definition 4.45—force two such morphisms to be equal, and so we dutifully equate them. When we are done, the result is again the free prop **Free**(G). Let's make this more formal.

First, we need the notion of a prop expression. Just as prop signatures are the analogue of the graphs used to present categories, prop expressions are the analogue of paths in these graphs.

Definition 5.30. Suppose we have a set G and functions $s, t : G \to \mathbb{N}$. We define a *G-generated prop expression*, or simply *expression* $e : m \to n$, where $m, n \in \mathbb{N}$, inductively as follows:

- The empty morphism $\mathrm{id}_0 : 0 \to 0$, the identity morphism $\mathrm{id}_1 : 1 \to 1$, and the symmetry $\sigma : 2 \to 2$ are expressions.[5]
- the generators $g \in G$ are expressions $g : s(g) \to t(g)$.
- if $\alpha : m \to n$ and $\beta : p \to q$ are expressions, then $\alpha + \beta : m + p \to n + q$ is an expression.
- if $\alpha : m \to n$ and $\beta : n \to p$ are expressions, then $\alpha \fatsemi \beta : m \to p$ is an expression.

We write $\mathrm{Expr}(G)$ for the set of expressions in G. If $e : m \to n$ is an expression, we refer to (m, n) as its *arity*.

Example 5.31. Let $G = \{f : 1 \to 1, g : 2 \to 2, h : 2 \to 1\}$. Then

- $\mathrm{id}_1 : 1 \to 1$,
- $f : 1 \to 1$,
- $f \fatsemi \mathrm{id}_1 : 1 \to 1$,
- $h + \mathrm{id}_1 : 3 \to 2$, and
- $(h + \mathrm{id}_1) \fatsemi \sigma \fatsemi g \fatsemi \sigma : 3 \to 2$

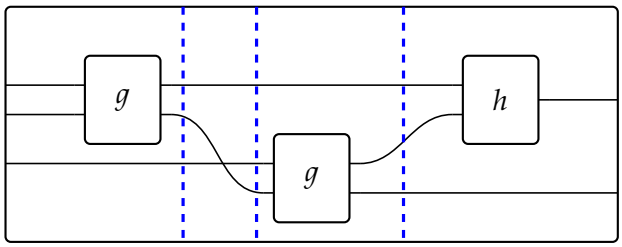
are all G-generated prop expressions.

Both G-labeled port graphs and G-generated prop expressions are ways to describe morphisms in the free prop **Free**(G). Note, however, that unlike for G-labeled port graphs, there may be two G-generated prop expressions that represent the same morphism. For example, we want to consider $f \fatsemi \mathrm{id}_1$ and f to be the same morphism, since the unitality axiom for categories says $f \fatsemi \mathrm{id}_1 = f$. Nonetheless, we only consider two G-generated prop expressions equal when some axiom from the definition of prop requires that they be so; again, the free prop is the *minimally-constrained* way to take G and obtain a prop.

Since both port graphs and prop expressions describe morphisms in **Free**(G), you might be wondering how to translate between them. Here's how to turn a port graph into a prop expression: imagine a vertical line moving through the port graph from

[5] One can think of σ as the "swap" icon $\times : 2 \to 2$

left to right. Whenever you see "action"—either a box or wires crossing—write down the sum (using +) of all the boxes g, all the symmetries σ, and all the wires id_1 in that column. Finally, compose all of those action columns. For example, in the picture below we see four action columns:



Here the result is $(g + \mathrm{id}_1) \fatsemi (\mathrm{id}_1 + \sigma) \fatsemi (\mathrm{id}_1 + g) \fatsemi (h + \mathrm{id}_1)$.

Exercise 5.32. Consider again the free prop on generators $G = \{f : 1 \to 1, g : 2 \to 2, h : 2 \to 1\}$. Draw a picture of $(f + \mathrm{id}_1 + \mathrm{id}_1) \fatsemi (\sigma + \mathrm{id}_1) \fatsemi (\mathrm{id}_1 + h) \fatsemi \sigma \fatsemi g$, where $\sigma : 2 \to 2$ is the symmetry map. ◊

Another way of describing when we should consider two prop expressions equal is to say that they are equal if and only if they represent the same port graph. In either case, these notions induce an equivalence relation on the set of prop expressions. To say that we consider these certain prop expressions equal is to say that the morphisms of the free prop on G are the G-generated prop expressions *quotiented* by this equivalence relation (see Definition 1.21).

5.2.5 Props via presentations

In Section 3.2.2 we saw that a presentation for a category, or database schema, consists of a graph together with imposed equations between paths. Similarly here, sometimes we want to construct a prop whose morphisms obey specific equations. But rather than mere paths, the things we want to equate are prop expressions as in Definition 5.30.

Rough Definition 5.33. A *presentation* (G, s, t, E) for a prop is a set G, functions $s, t : G \to \mathbb{N}$, and a set $E \subseteq \mathrm{Expr}(G) \times \mathrm{Expr}(G)$ of pairs of G-generated prop expressions, such that e_1 and e_2 have the same arity for each $(e_1, e_2) \in E$. We refer to G as the set of generators and to E as the set of *equations* in the presentation.[6]

The prop $\mathcal{G}$ *presented* by the presentation (G, s, t, E) is the prop whose morphisms are elements in $\mathrm{Expr}(G)$, quotiented by both the equations $e_1 = e_2$ where $(e_1, e_2) \in E$, and by the axioms of symmetric strict monoidal categories.

Remark 5.34. Given a presentation (G, s, t, E), it can be shown that the prop $\mathcal{G}$ has a universal property in terms of "maps out." Namely prop functors from $\mathcal{G}$ to any

[6] Elements of E, which we call equations, are traditionally called "relations." We think of $(e_1, e_2) \in E$ as standing for the equation $e_1 = e_2$, as this will be forced soon.

other prop $\mathcal{C}$ are in one-to-one correspondence with functions f from G to the set of morphisms in $\mathcal{C}$ such that

- for all $g \in G$, $f(g)$ is a morphism $s(g) \to t(g)$, and
- for all $(e_1, e_2) \in E$, we have that $f(e_1) = f(e_2)$ in $\mathcal{C}$, where $f(e)$ denotes the morphism in $\mathcal{C}$ obtained by applying f to each generators in the expression e, and then composing the result in $\mathcal{C}$.

Exercise 5.35. Is it the case that the free prop on generators (G, s, t), defined in Definition 5.25, is the same thing as the prop presented by $(G, s, t, \varnothing)$, having no relations, as defined in Definition 5.33? Or is there a subtle difference somehow? ◊

5.3 Simplified signal flow graphs

We now return to signal flow graphs, expressing them in terms of props. We will discuss a simplified form without feedback (the only sort we have discussed so far), and then extend to the usual form of signal flow graphs in Section 5.4.3. But before we can do that, we must say what we mean by signals; this gets us into the algebraic structure of "rigs." We will get to signal flow graphs in Section 5.3.2.

5.3.1 Rigs

Signals can be amplified, and they can be added. Adding and amplification interact via a distributive law, as follows: if we add two signals, and then amplify them by some amount a, it should be the same as amplifying the two signals separately by a, then adding the results.

We can think of all the possible amplifications as forming a structure called a rig,[7] defined as follows.

Definition 5.36. A *rig* is a tuple $(R, 0, +, 1, *)$, where R is a set, $0, 1 \in R$ are elements, and $+, *\colon R \times R \to R$ are functions, such that

(a) $(R, +, 0)$ is a commutative monoid,
(b) $(R, *, 1)$ is a monoid,[8] and
(c) $a * (b + c) = a * b + a * c$ and $(a + b) * c = a * c + b * c$ for all $a, b, c \in R$.
(d) $a * 0 = 0 = 0 * a$ for all $a \in R$.

We have already encountered many examples of rigs.

Example 5.37. The natural numbers form a rig $(\mathbb{N}, 0, +, 1, *)$.

[7] Rigs are also known as *semi-rings*.

[8] Note that we did not demand that $(R, *, 1)$ be commutative; we will see a naturally-arising example where it is not commutative in Example 5.40.

Example 5.38. The Booleans form a rig $(\mathbb{B}, \texttt{false}, \vee, \texttt{true}, \wedge)$.

Example 5.39. Any quantale $\mathcal{V} = (V, \leq, I, \otimes)$ determines a rig $(V, 0, \vee, I, \otimes)$, where $0 = \bigvee \varnothing$ is the empty join. See Definition 2.79.

Example 5.40. If R is a rig and $n \in \mathbb{N}$ is any natural number, then the set $\mathrm{Mat}_n(R)$ of $(n \times n)$-matrices in R forms a rig. A matrix $M \in \mathrm{Mat}_n(R)$ is a function $M \colon \underline{n} \times \underline{n} \to R$. Addition $M + N$ of matrices is given by $(M + N)(i, j) \coloneqq M(i, j) + N(i, j)$ and multiplication $M * N$ is given by $(M * N)(i, j) \coloneqq \sum_{k \in \underline{n}} M(i, k) * N(k, j)$. The 0-matrix is $0(i, j) \coloneqq 0$ for all $i, j \in \underline{n}$. Note that $\mathrm{Mat}_n(R)$ is generally not commutative.

Exercise 5.41.

1. We said in Example 5.40 that for any rig R, the set $\mathrm{Mat}_n(R)$ forms a rig. What is its multiplicative identity $1 \in \mathrm{Mat}_n(R)$?
2. We also said that $\mathrm{Mat}_n(R)$ is generally not commutative. Pick an n and show that that $\mathrm{Mat}_n(\mathbb{N})$ is not commutative, where $\mathbb{N}$ is as in Example 5.37. ◊

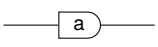
The following is an example for readers who are familiar with the algebraic structure known as "rings."

Example 5.42. Any ring forms a rig. In particular, the real numbers $(\mathbb{R}, 0, +, 1, *)$ are a rig. The difference between a ring and rig is that a ring, in addition to all the properties of a rig, must also have additive inverses, or *negatives*. A common mnemonic is that a rig is a ri**n**g without **n**egatives.

5.3.2 The iconography of signal flow graphs

A signal flow graph is supposed to keep track of the amplification, by elements of a rig R, to which signals are subjected. While not strictly necessary,[9] we will assume the signals themselves are elements of the same rig R. We refer to elements of R as *signals* for the time being.

Amplification of a signal by some value $a \in R$ is simply depicted like so:



(scalar mult.)

We interpret the above icon as a depicting a system where a signal enters on the left-hand wire, is multiplied by a, and is output on the right-hand wire.

[9] The necessary requirement for the material below to make sense is that the signals take values in an *R-module* M. We will not discuss this here, keeping to the simpler requirement that $M = R$.

What is more interesting than just a single signal amplification, however, is the interaction of signals. There are four other important icons in signal flow graphs.

Let's go through them one by one. The first two are old friends from Chapter 2: copy and discard.

(copy)

We interpret this diagram as taking in an input signal on the left, and outputting that same value to both wires on the right. It is basically the "copy" operation from Section 2.2.3.

Next, we have the ability to discard signals.

(discard)

This takes in any signal, and outputs nothing. It is basically the "waste" operation from Section 2.2.3.

Next, we have the ability to add signals.

(add, +)

This takes the two input signals and adds them, to produce a single output signal.

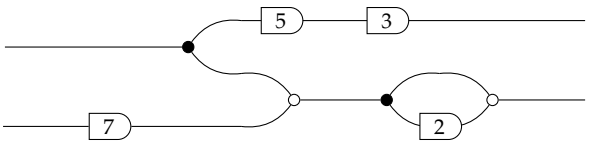
Finally, we have the zero signal.

(zero, 0)

This has no inputs, but always outputs the 0 element of the rig.

Using these icons, we can build more complex signal flow graphs. To compute the operation performed by a signal flow graph we simply trace the paths with the above interpretations, plugging outputs of one icon into the inputs of the next icon.

For example, consider the rig $R = \mathbb{N}$ from Example 5.37, where the scalars are the natural numbers. Recall the signal flow graph from Eq. (5.1) in the introduction:

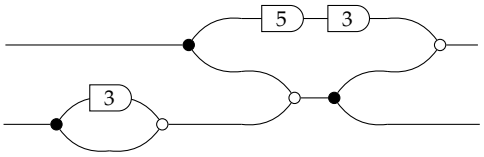


As we explained, this takes in two input signals x and y, and returns two output signals $a = 15x$ and $b = 3x + 21y$.

In addition to tracing the processing of the values as they move forward through the graph, we can also calculate these values by summing over paths. More explicitly, to get the contribution of a given input wire to a given output wire, we take the sum, over all paths p joining the wires, of the total amplification along that path.

So, for example, there is one path from the top input to the top output. On this path, the signal is first copied, which does not affect its value, then amplified by 5, and finally amplified by 3. Thus, if x is the first input signal, then this contributes $15x$ to the first output. Since there is no path from the bottom input to the top output (one is not allowed to traverse paths backwards), the signal at the first output is exactly $15x$. Both inputs contribute to the bottom output. In fact, each input contributes in two ways, as there are two paths to it from each input. The top input thus contributes $3x = x + 2x$, whereas the bottom input, passing through an additional $*7$ amplification, contributes $21y$.

Exercise 5.43. The following flow graph takes in two natural numbers x and y



and produces two output signals. What are they? ◊

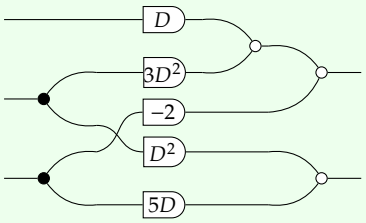
Example 5.44. This example is for those who have some familiarity with differential equations. A linear system of differential equations provides a simple way to specify the movement of a particle. For example, consider a particle whose position (x, y, z) in 3-dimensional space is determined by the following equations:

$$\begin{aligned} \dot{x} + 3\ddot{y} - 2z &= 0 \\ \ddot{y} + 5\dot{z} &= 0 \end{aligned}$$

Using what is known as the Laplace transform, one can convert this into a linear system involving a formal variable D, which stands for "differentiate." Then the system becomes

$$\begin{aligned} Dx + 3D^2y - 2z &= 0 \\ D^2y + 5Dz &= 0 \end{aligned}$$

which can be represented by the signal flow graph



Signal flow graphs as morphisms in a free prop. We can formally define simplified signal flow graphs using props.

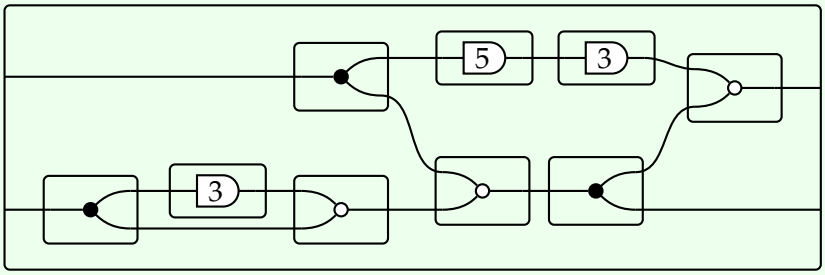
Definition 5.45. Let R be a rig (see Definition 5.36). Consider the set

$$G_R := \left\{ \right\} \cup \left\{ \; | \; a \in R \right\},$$

and let $s, t \colon G_R \to \mathbb{N}$ be given by the number of dangling wires on the left and right of the generator icon respectively. A *simplified signal flow graph* is a morphism in the free prop $\mathbf{Free}(G_R)$ on this set G_R of generators. We define $\mathbf{SFG}_R := \mathbf{Free}(G_R)$.

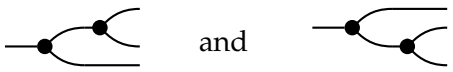
For now we'll drop the term 'simplified', since these are the only sort of signal flow graph we know. We'll return to signal flow graphs in their full glory—i.e. including feedback—in Section 5.4.3.

Example 5.46. To be more in line with our representations of both wiring diagrams and port graphs, morphisms in $\mathbf{Free}(G_R)$ should be drawn slightly differently. For example, technically the signal flow graph from Exercise 5.43 should be drawn as follows:



because we said we would label boxes with the elements of G. But it is easier on the eye to draw remove the boxes and just look at the icons inside as in Exercise 5.43, and so we'll draw our diagrams in that fashion.

More importantly, props provide language to understand the semantics of signal flow graphs. Although the signal flow graphs themselves are free props, their semantics—their meaning in our model of signals flowing—will arise when we add equations to our props, as in Definition 5.33. These equations will tell us when two signal flow graphs act the same way on signals. For example,



(5.47)

both express the same behavior: a single input signal is copied twice so that three identical copies of the input signal are output.

If two signal flow graphs S, T are almost the same, with the one exception being that somewhere we replace the left-hand side of Eq. (5.47) with the right-hand side, then S and T have the same behavior. But there are other replacements we could make to a signal flow graph that do not change its behavior. Our next goal is to find a complete description of these replacements.

5.3.3 The prop of matrices over a rig

Signal flow graphs are closely related to matrices. In previous chapters we showed how a matrix with values in a quantale $\mathcal{V}$—a closed monoidal preorder with all joins—represents a system of interrelated points and connections between them, such as a profunctor. The quantale gave us the structure and axioms we needed in order for matrix multiplication to work properly. But we know from Example 5.39 that quantales are examples of rigs, and in fact matrix multiplication makes sense in any rig R. In Example 5.40, we explained that the set $\mathrm{Mat}_n(R)$ of $(n \times n)$-matrices in R can naturally be assembled into a rig, for any fixed choice of $n \in \mathbb{N}$. But what if we want to do better, and assemble *all* matrices into a single algebraic structure? The result is a prop!

An *$(m \times n)$-matrix M with values in R* is a function $M \colon (\underline{m} \times \underline{n}) \to R$. Given an $(m \times n)$-matrix M and an $(n \times p)$-matrix N, their *composite* is the $(m \times p)$-matrix $M \fatsemi N$ defined as follows for any $a \in \underline{m}$ and $c \in \underline{p}$:

$$M \fatsemi N(a, c) \coloneqq \sum_{b \in \underline{n}} M(a, b) \times N(b, c), \tag{5.48}$$

Here the $\sum_{b \in \underline{n}}$ just means repeated addition (using the rig R's + operation), as usual.

Remark 5.49. Conventionally, one generally considers a matrix A acting on a vector v by multiplication in the order Av, where v is a column vector. In keeping with our composition convention, we use the opposite order, $v \fatsemi A$, where v is a row vector. See for example Eq. (5.52) for when this is implicitly used.

Definition 5.50. Let R be a rig. We define the *prop of R-matrices*, denoted $\mathbf{Mat}(R)$, to be the prop whose morphisms $m \to n$ are the $(m \times n)$-matrices with values in R. Composition of morphisms is given by matrix multiplication as in Eq. (5.48). The monoidal product is given by the direct sum of matrices: given matrices $A \colon m \to n$ and $b \colon p \to q$, we define $A + B \colon m + p \to n + q$ to be the block matrix

$$\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$$

where each 0 represents a matrix of zeros of the appropriate dimension ($m \times q$ and $n \times p$). We refer to any combination of multiplication and direct sum as a *interconnection* of matrices.

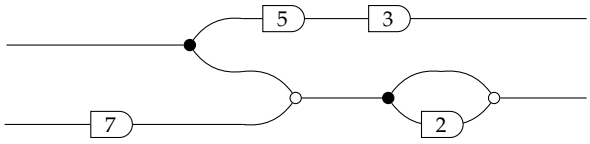
Exercise 5.51. Let A and B be the following matrices with values in $\mathbb{N}$:

$$A = \begin{pmatrix} 3 & 3 & 1 \\ 2 & 0 & 4 \end{pmatrix} \qquad\qquad B = \begin{pmatrix} 2 & 5 & 6 & 1 \end{pmatrix}.$$

What is the direct sum matrix $A + B$? ◊

5.3.4 Turning signal flow graphs into matrices

Let's now consider more carefully what we mean when we talk about the meaning, or *semantics*, of each signal flow graph. We'll use matrices.



In the examples like the above (copied from Eq. (5.1)), the signals emanating from output wires, say a and b, are given by certain sums of amplified input values, say x and y. If we can only measure the input and output signals, and care nothing for what happens in between, then each signal flow graph may as well be reduced to a matrix of amplifications. We can represent the signal flow graph of Eq. (5.1) by either the matrix on the left (for more detail) or the matrix on the right if the labels are clear from context:

	a	b
x	15	3
y	0	21

$$\begin{pmatrix} 15 & 3 \\ 0 & 21 \end{pmatrix}$$

Every signal flow graph can be interpreted as a matrix. The generators G_R from Definition 5.45 are shown again in the table below, where each is interpreted as a matrix. For example, we interpret amplification by $a \in R$ as the 1×1 matrix $(a)\colon 1 \to 1$: it is an operation that takes an input $x \in R$ and returns $a * x$. Similarly, we can interpret  as the 2×1 matrix $\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$: it is an operation that takes a row vector consisting of two inputs, x and y, and returns $x + y$. Here is a table showing the interpretation of each generator.

generator	icon	matrix	arity
amplify by $a \in R$		(a)	$1 \to 1$
add		$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$2 \to 1$
zero		$()$	$0 \to 1$
copy		$(1 \quad 1)$	$1 \to 2$
discard		$()$	$1 \to 0$

(5.52)

Note that both zero and discard are represented by empty matrices, but of differing dimensions. In linear algebra it is unusual to consider matrices of the form $0 \times n$ or $n \times 0$ for various n to be different, but they can be kept distinct for bookkeeping purposes: you can multiply a 0×3 matrix by a $3 \times n$ matrix for any n, but you can not multiply it by a $2 \times n$ matrix.

Since signal flow graphs are morphisms in a free prop, the table in (5.52) is enough to show that we can interpret any signal flow diagram as a matrix.

Theorem 5.53. There is a prop functor $S\colon \mathbf{SFG}_R \to \mathbf{Mat}(R)$ that sends the generators $g \in G$ icons to the matrices as described in Table 5.52.

Proof. This follows immediately from the universal property of free props, Remark 5.34. □

We have now constructed a matrix $S(g)$ from any signal flow graph g. But how can we produce this matrix explicitly? Both for the example signal flow graph in Eq. (5.1) and for the generators in Definition 5.45, the associated matrix has dimension $m \times n$, where m is the number of inputs and n the number of outputs, with (i, j)th entry describing the amplification of the ith input that contributes to the jth output. This is how one would hope or expect the functor S to work in general; but does it? We have used a big hammer—the universal property of free constructions—to obtain our functor S. Our next goal is to check that it works in the expected way. Doing so is a matter of using induction over the set of prop expressions, as we now see.[10]

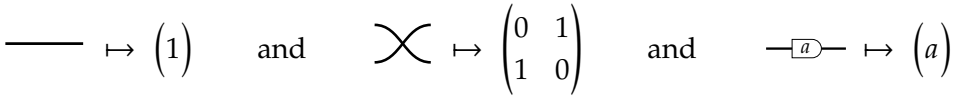
Proposition 5.54. Let g be a signal flow graph with m inputs and n outputs. The matrix $S(g)$ is the $(m \times n)$-matrix whose (i, j)-entry describes the amplification of the ith input that contributes to the jth output.

Proof. Recall from Definition 5.30 that an arbitrary G_R-generated prop expression is built from the morphisms $\mathrm{id}_0\colon 0 \to 0$, $\mathrm{id}_1\colon 1 \to 1$, $\sigma\colon 2 \to 2$, and the generators in G_R, using the following two rules:

- if $\alpha\colon m \to n$ and $\beta\colon p \to q$ are expressions, then $(\alpha + \beta)\colon (m + p) \to (n + q)$ is an expression.
- if $\alpha\colon m \to n$ and $\beta\colon n \to p$ are expressions, then $\alpha \fatsemi \beta\colon m \to p$ is an expression.

S is a prop functor by Theorem 5.53, which by Definition 5.11 must preserve identities, compositions, monoidal products, and symmetries. We first show that the proposition is true when g is equal to id_0, id_1, and σ.

The empty signal flow graph $\mathrm{id}_0\colon 0 \to 0$ must be sent to the unique (empty) matrix $()\colon 0 \to 0$. The morphisms id_1, σ, and $a \in R$ map to the identity matrix, the swap matrix, and the scalar matrix (a) respectively:

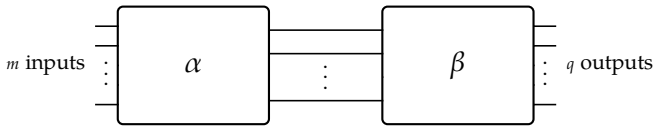


In each case, the (i, j)-entry gives the amplification of the ith input to the jth output.

It remains to show that if the proposition holds for $\alpha\colon m \to n$ and $\beta\colon p \to q$, then it holds for (i) $\alpha \fatsemi \beta$ (when $n = p$) and for (ii) $\alpha + \beta$ (in general).

[10] Mathematical induction is a formal proof technique that can be thought of like a domino rally: if you knock over all the starting dominoes, and you're sure that each domino will be knocked down if its predecessors are, then you're sure every domino will eventually fall. If you want more rigor, or you want to understand the proof of Proposition 5.54 as a genuine case of induction, ask a friendly neighborhood mathematician!

To prove (i), consider the following picture of $\alpha \fatsemi \beta$:

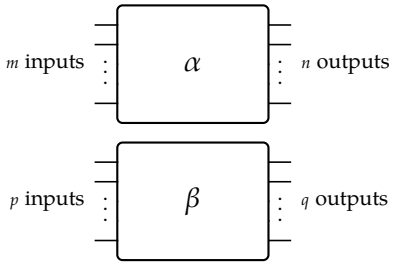


Here $\alpha \colon m \to n$ and $\beta \colon n \to q$ are signal flow graphs, assumed to obey the proposition. Consider the ith input and kth output of $\alpha \fatsemi \beta$; we'll just call these i and k. We want to show that the amplification that i contributes to k is the sum—over all paths from i to k—of the amplification along that path. So let's also fix some $j \in \underline{n}$, and consider paths from i to k that run through j. By distributivity of the rig R, the total amplification from i to k through j is the total amplification over all paths from i to j times the total amplication over all paths from j to k. Since all paths from i to k must run through some jth output of α/input of β, the amplification that i contributes to k is

$$\sum_{j \in \underline{n}} \alpha(i, j) * \beta(j, k).$$

This is exactly the formula for matrix multiplication, which is composition $S(\alpha) \fatsemi S(\beta)$ in the prop $\mathbf{Mat}(R)$; see Definition 5.50. So $\alpha \fatsemi \beta$ obeys the proposition when α and β do.

Proving (ii) is more straightforward. The monoidal product $\alpha + \beta$ of signal flow graphs looks like this:



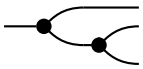
No new paths are created; the only change is to reindex the inputs and outputs. In particular, the ith input of α is the ith input of $\alpha + \beta$, the jth output of α is the jth output of $\alpha + \beta$, the ith input of β is the $(m + i)$th output of $\alpha + \beta$, and the jth output of β is the $(n + j)$th output of $\alpha + \beta$. This means that the matrix with (i, j)th entry describing the amplification of the ith input that contributes to the jth output is $S(\alpha) + S(\beta) = S(\alpha + \beta)$, as in Definition 5.50. This proves the proposition. □

Exercise 5.55.

1. What matrix does the signal flow graph

represent?

2. What about the signal flow graph



3. Are they equal? ◊

5.3.5 The idea of functorial semantics

Let's pause for a moment to reflect on what we have just learned. First, signal flow diagrams are the morphisms in a prop. This means we have two special operations we can do to form new signal flow diagrams from old, namely composition (combining in series) and monoidal product (combining in parallel). We might think of this as specifying a 'grammar' or 'syntax' for signal flow diagrams.

As a language, signal flow graphs have not only syntax but also semantics: each signal flow diagram can be interpreted as a matrix. Moreover, matrices have the same grammatical structure: they form a prop, and we can construct new matrices from old using composition and monoidal product. In Theorem 5.53 we completed this picture by showing that semantic interpretation is a prop functor between the prop of signal flow graphs and the prop of matrices. Thus we say that matrices give *functorial semantics* for signal flow diagrams.

Functorial semantics is a key manifestation of compositionality. It says that the matrix meaning $S(g)$ for a big signal flow graph g can be computed by:

1. splitting g up into little pieces,
2. computing the very simple matrices for each piece, and
3. using matrix multiplication and direct sum to put the pieces back together to obtain the desired meaning, $S(g)$.

This functoriality is useful in practice, for example in speeding up computation of the semantics of signal flow graphs: for large signal flow graphs, composing matrices is much faster than tracing paths.

5.4 Graphical linear algebra

In this section we will begin to develop something called graphical linear algebra, which extends the ideas above. This formalism is actually quite powerful. For example, with it we can easily and *graphically* prove certain conjectures from control theory that, although they were eventually solved, required fairly elaborate matrix algebra arguments [FSR16].

5.4.1 A presentation of $\mathbf{Mat}(R)$

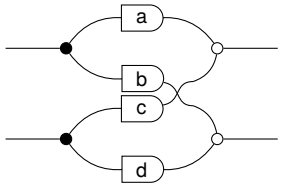
Let R be a rig, as defined in Definition 5.36. The main theorem of the previous section, Theorem 5.53, provided a functor $S\colon \mathbf{SFG}_R \to \mathbf{Mat}(R)$ that converts any signal flow

graph into a matrix. Next we show that S is "full": that any matrix can be represented by a signal flow graph.

Proposition 5.56. Given any matrix $M \in \mathbf{Mat}(R)$, there exists a signal flow graph $g \in \mathbf{SFG}_R$ such that such that $S(g) = M$.

Proof sketch. Let $M \in \mathbf{Mat}(R)$ be an $(m \times n)$-matrix. We want a signal flow graph g such that $S(g) = M$. In particular, to compute $S(g)(i, j)$, we know that we can simply compute the amplification that the ith input contributes to the jth output. The key idea then is to construct g so that there is exactly one path from ith input to the jth output, and that this path has exactly one scalar multiplication icon, namely $M(i, j)$.

The general construction is a little technical (see Exercise 5.59), but the idea is clear from just considering the case of 2×2-matrices. Suppose M is the 2×2-matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$. Then we define g to be the signal flow graph



(5.57)

Tracing paths, it is easy to see that $S(g) = M$. Note that g is the composite of four layers, each layer respectively a monoidal product of (i) copy and discard maps, (ii) scalar multiplications, (iii) swaps and identities, (iv) addition and zero maps.

For the general case, see Exercise 5.59. □

Exercise 5.58. Draw signal flow graphs that represent the following matrices:

$$1.\ \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} \qquad 2.\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad 3.\ \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

◊

Exercise 5.59. Write down a detailed proof of Proposition 5.56. Suppose M is an $m \times n$-matrix. Follow the idea of the (2×2)-case in Eq. (5.57), and construct the signal flow graph g—having m inputs and n outputs—as the composite of four layers, respectively comprising (i) copy and discard maps, (ii) scalars, (iii) swaps and identities, (iv) addition and zero maps. ◊

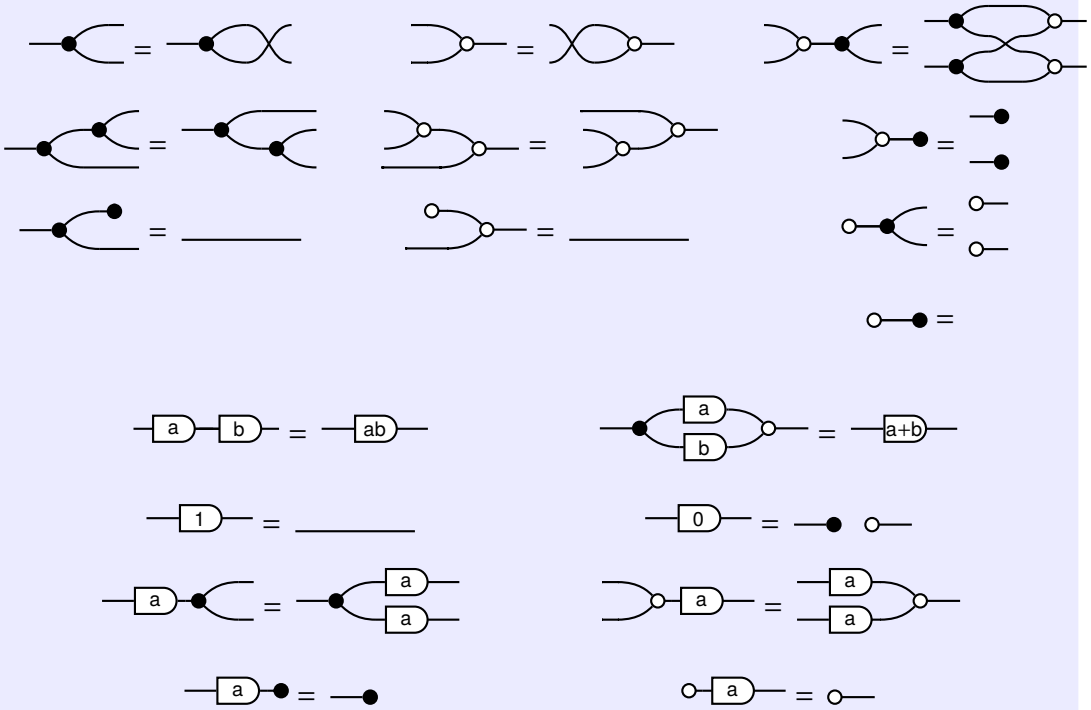
We can also use Proposition 5.56 and its proof to give a presentation of $\mathbf{Mat}(R)$, which was defined in Definition 5.50.

Theorem 5.60. The prop **Mat**(R) is isomorphic to the prop with the following presentation. The set of generators is the set

$$G_R := \left\{ \; , \; , \; , \; \right\} \cup \left\{ \; \boxed{a} \; \middle| \; a \in R \right\},$$

the same as the set of generators for $\mathbf{SFG}_R$; see Definition 5.45.

We have the following equations for any $a, b \in R$:



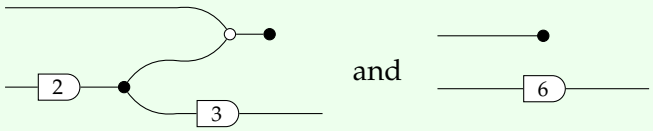
Proof. The key idea is that these equations are sufficient to rewrite any G_R-generated prop expression into a normal form—the one used in the proof of Proposition 5.56—with all the black nodes to the left, all the white nodes to the right, and all the scalars in the middle. This is enough to show the equality of any two expressions that represent the same matrix. Details can be found in [BE15] or [BS17]. □

Sound and complete presentation of matrices. Once you get used to it, Theorem 5.60 provides an intuitive, visual way to reason about matrices. Indeed, the theorem implies two signal flow graphs represent the same matrix if and only if one can be turned into the other by local application of the above equations and the prop axioms.

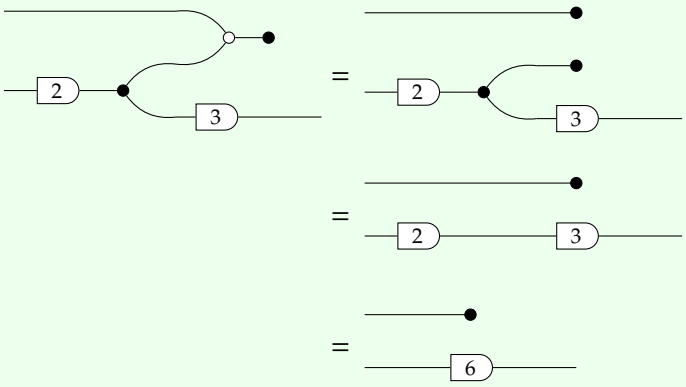
The fact that you can prove two SFGs to be the same by using only graphical rules can be stated in the jargon of logic: we say that the graphical rules provide a *sound and complete reasoning system*. To be more specific, *sound* refers to the forward direction of the above statement: two signal flow graphs represent the same matrix if one can be turned into the other using the given rules. *Complete* refers to the reverse direction: if

two signal flow graphs represent the same matrix, then we can convert one into the other using the equations of Theorem 5.60.

Example 5.61. Both of the signal flow graphs below represent the same matrix, $\binom{0}{6}$:



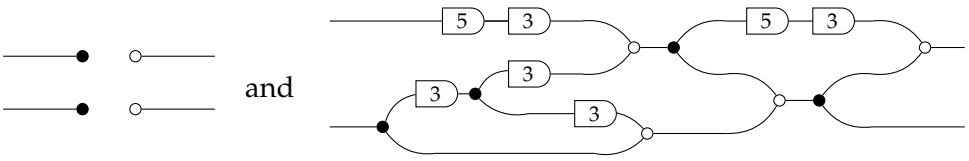
This means that one can be transformed into the other by using only the equations from Theorem 5.60. Indeed, here



Exercise 5.62.

1. For each matrix in Exercise 5.58, draw another signal flow graph that represents that matrix.
2. Using the above equations and the prop axioms, prove that the two signal flow graphs represent the same matrix. ◊

Exercise 5.63. Consider the signal flow graphs



(5.64)

1. Let $R = (\mathbb{N}, 0, +, 1, *)$. By examining the presentation of $\mathbf{Mat}(R)$ in Theorem 5.60, and without computing the matrices that the two signal flow graphs in Eq. (5.64) represent, prove that they do *not* represent the same matrix.
2. Now suppose the rig is $R = \mathbb{N}/3\mathbb{N}$; if you do not know what this means, just replace all 3's with 0's in the right-hand diagram of Eq. (5.64). Find what you would call a minimal representation of this diagram, using the presentation in Theorem 5.60. ◊

5.4.2 Aside: monoid objects in a monoidal category

Various subsets of the equations in Theorem 5.60 encode structures that are familiar from many other parts of mathematics, e.g. representation theory. For example one can find the axioms for (co)monoids, (co)monoid homomorphisms, Frobenius algebras, and (with a little rearranging) Hopf algebras, sitting inside this collection. The first example, the notion of monoids, is particularly familiar to us by now, so we briefly discuss it below, both in algebraic terms (Definition 5.65) and in diagrammatic terms (Example 5.68).

Definition 5.65. A *monoid object* (M, μ, η) in a symmetric monoidal category $(\mathcal{C}, I, \otimes)$ is an object M of $\mathcal{C}$ together with morphisms $\mu\colon M \otimes M \to M$ and $\eta\colon I \to M$ such that

(a) $(\mu \otimes \mathrm{id}) \fatsemi \mu = (\mathrm{id} \otimes \mu) \fatsemi \mu$ and

(b) $(\eta \otimes \mathrm{id}) \fatsemi \mu = \mathrm{id} = (\mathrm{id} \otimes \eta) \fatsemi \mu$.

A *commutative monoid object* is a monoid object that further obeys

(c) $\sigma_{M,M} \fatsemi \mu = \mu$.

where $\sigma_{M,M}$ is the swap map on M in $\mathcal{C}$. We often denote it simply by σ.

Monoid objects are so-named because they are an abstraction of the usual concept of monoid.

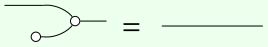
Example 5.66. A monoid object in $(\mathbf{Set}, 1, \times)$ is just a regular old monoid, as defined in Example 2.6; see also Example 3.13. That is, it is a set M, a function $\mu\colon M \times M \to M$, which we denote by infix notation $*$, and an element $\eta(1) \in M$, which we denote by e, satisfying $(a * b) * c = a * (b * c)$ and $a * e = a = e * a$.

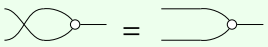
Exercise 5.67. Consider the set $\mathbb{R}$ of real numbers.

1. Show that if $\mu\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is defined by $\mu(a, b) = a * b$ and if $\eta \in \mathbb{R}$ is defined to be $\eta = 1$, then $(\mathbb{R}, *, 1)$ satisfies all three conditions of Definition 5.65.
2. Show that if $\mu\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is defined by $\mu(a, b) = a + b$ and if $\eta \in \mathbb{R}$ is defined to be $\eta = 0$, then $(\mathbb{R}, +, 0)$ satisfies all three conditions of Definition 5.65. ◊

Example 5.68. Graphically, we can depict $\mu =$  and $\eta =$  . Then axioms (a), (b), and (c) from Definition 5.65 become:

(a)

(b) 

(c) 

All three of these are found in Theorem 5.60. Thus we can immediately conclude the following: the triple $(1,$, $)$ is a commutative monoid object in the prop $\mathbf{Mat}(R)$.

Exercise 5.69. For any rig R, there is a functor $U\colon \mathbf{Mat}(R) \to \mathbf{Set}$, sending the object $n \in \mathbb{N}$ to the set R^n, and sending a morphism (matrix) $M\colon m \to n$ to the function $R^m \to R^n$ given by vector-matrix multiplication.

Recall that in $\mathbf{Mat}(R)$, the monoidal unit is 0 and the monoidal product is +, because it is a prop. Recall also that in (the usual monoidal structure on) **Set**, the monoidal unit is $\{1\}$, a set with one element, and the monoidal product is $\times$ (see Example 4.49).

1. Check that the functor $U\colon \mathbf{Mat}(R) \to \mathbf{Set}$, defined above, preserves the monoidal unit and the monoidal product.
2. Show that if (M, μ, η) is a monoid object in $\mathbf{Mat}(R)$ then $(U(M), U(\mu), U(\eta))$ is a monoid object in **Set**. (This works for any monoidal functor—which we will define in Definition 6.68—not just for U in particular.)
3. In Example 5.68, we said that the triple $(1, \rightarrowtail, \twoheadleftarrow)$ is a commutative monoid object in the prop $\mathbf{Mat}(R)$. If $R = \mathbb{R}$ is the rig of real numbers, this means that we have a monoid structure on the set $\mathbb{R}$. But in Exercise 5.67 we gave two such monoid structures. Which one is it? ◊

Example 5.70. The triple $(1, \rightarrowtail, \twoheadrightarrow)$ in $\mathbf{Mat}(R)$ forms a commutative monoid object in $\mathbf{Mat}(R)^{\mathrm{op}}$. We hence also say that $(1, \rightarrowtail, \twoheadrightarrow)$ forms a *co-commutative comonoid object* in $\mathbf{Mat}(R)$.

Example 5.71. A *symmetric strict monoidal category*, is just a commutative monoid object in $(\mathbf{Cat}, \times, \mathbf{1})$. We will unpack this in Section 6.4.1.

Example 5.72. A symmetric monoidal preorder, which we defined in Definition 2.2, is just a commutative monoid object in the symmetric monoidal category $(\mathbf{Preord}, \times, \mathbf{1})$ of preorders and monotone maps.

Example 5.73. For those who know what tensor products of commutative monoids are (or can guess): A rig is a monoid object in the symmetric monoidal category $(\mathbf{CMon}, \otimes, \mathbb{N})$ of commutative monoids with tensor product.

Remark 5.74. If we present a prop $\mathcal{M}$ using two generators $\mu\colon 2 \to 1$ and $\eta\colon 0 \to 1$, and the three equations from Definition 5.65, we could call it 'the theory of monoids in monoidal categories.' This means that in any monoidal category $\mathcal{C}$, the monoid objects in $\mathcal{C}$ correspond to strict monoidal functors $\mathcal{M} \to \mathcal{C}$. This sort of idea leads to the study of algebraic theories, due to Bill Lawvere and extended by many others; see Section 5.5.

5.4.3 Signal flow graphs: feedback and more

At this point in the story, we have seen that every signal flow graph represents a matrix, and this gives us a new way of reasoning about matrices. This is just the beginning of a beautiful tale, one not only of graphical matrices, but of graphical *linear algebra*. We close this chapter with some brief hints at how the story continues.

The pictoral nature of signal flow graphs invites us to play with them. While we normally draw the copy icon like so,  , we could just as easily reverse it and draw an icon . What might it mean? Let's think again about the semantics of flow graphs.

The behavioral approach. A signal flow graph $g\colon m \to n$ takes an input $x \in R^m$ and gives an output $y \in R^n$. In fact, since this is all we care about, we might just think about representing a signal flow graph g as describing a set of input and output pairs (x, y). We'll call this set the *behavior* of g and denote it $\mathsf{B}(g) \subseteq R^m \times R^n$. For example, the 'copy' flow graph

sends the input 1 to the output $(1, 1)$, so we consider $(1, (1, 1))$ to be an element of copy-behavior. Similarly, $(x, (x, x))$ is copy behavior for every $x \in R$, thus we have

$$\mathsf{B}(\text{}) = \{(x, (x, x)) \mid x \in R\}.$$

In the abstract, the signal flow graph $g\colon m \to n$ has the behavior

$$\mathsf{B}(g) = \left\{\big(x, S(g)(x)\big) \mid x \in R^m\right\} \subseteq R^m \times R^n. \tag{5.75}$$

Mirror image of an icon. The above behavioral perspective provides a clue about how to interpret the mirror images of the diagrams discussed above. Reversing an icon $g\colon m \to n$ exchanges the inputs with the outputs, so if we denote this reversed icon by g^{op}, we must have $g^{\mathrm{op}}\colon n \to m$. Thus if $\mathsf{B}(g) \subseteq R^m \times R^n$ then we need $\mathsf{B}(g^{\mathrm{op}}) \subseteq R^n \times R^m$. One simple way to do this is to replace each (a, b) with (b, a), so we would have

$$\mathsf{B}(g^{\mathrm{op}}) := \left\{\big(S(g)(x), x\big) \mid x \in R^m\right\} \subseteq R^n \times R^m. \tag{5.76}$$

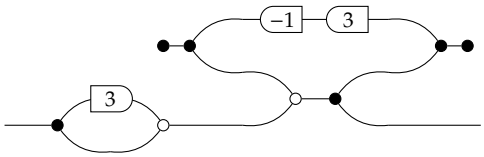
This is called the *transposed relation*.

Exercise 5.77.

1. What is the behavior $\mathsf{B}(\text{})$ of the reversed addition icon : $1 \to 2$?
2. What is the behavior $\mathsf{B}(\text{})$ of the reversed copy icon, : $2 \to 1$? ◊

Eqs. (5.75) and (5.76) give us formulas for interpreting signal flow graphs and their mirror images. But this would easily lead to disappointment, if we couldn't combine the two directions behaviorally; luckily we can.

Combining directions. What should the behavior be for a diagram such as the following:



Let's formalize our thoughts a bit and begin by thinking about behaviors. The behavior of a signal flow graph $m \to n$ is a subset $B \subseteq R^m \times R^n$, i.e. a relation. Why not try to construct a prop where the morphisms $m \to n$ are relations?

We'll need to know how to compose and take monoidal products of relations. And if we want this prop of relations to contain the old prop $\mathbf{Mat}(R)$, we need the new compositions and monoidal products to generalize the old ones in $\mathbf{Mat}(R)$. Given signal flow graphs with matrices $M\colon m \to n$ and $N\colon n \to p$, we see that their behaviors are the relations $B_1 := \{(x, Mx) \mid x \in R^m\}$ and $B_2 := \{(y, Ny) \mid y \in R^n\}$, while the behavior of $M \fatsemi N$ is the relation $\{(x, x \fatsemi M \fatsemi N) \mid x \in R^m\}$. This is a case of relation composition. Given relations $B_1 \subseteq R^m \times R^n$ and $B_2 \subseteq R^n \times R^p$, their composite $B_1 \fatsemi B_2 \subseteq R^m \times R^p$ is given by

$$B_1 \fatsemi B_2 := \{(x, z) \mid \text{there exists } y \in R^n \text{ such that } (x, y) \in B_1 \text{ and } (y, z) \in B_2\}. \tag{5.78}$$

We shall use this as the general definition for composing two behaviors.

Definition 5.79. Let R be a rig. We define the prop $\mathbf{Rel}_R$ of R-relations to have subsets $B \subseteq R^m \times R^n$ as morphisms. These are composed by the composition rule from Eq. (5.78), and we take the product of two sets to form their monoidal product.

Exercise 5.80. In Definition 5.79 we went quickly through monoidal products + in the prop $\mathbf{Rel}_R$. If $B \subseteq R^m \times R^n$ and $C \subseteq R^p \times R^q$ are morphisms in $\mathbf{Rel}_R$, write down $B + C$ in set-notation. ◊

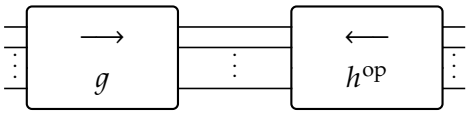
(No-longer simplified) signal flow graphs. Recall that above, e.g. in Definition 5.45, we wrote G_R for the set of generators of signal flow graphs. In Section 5.4.3, we wrote g^{op} for the mirror image of g, for each $g \in G_R$. So let's write $G_R^{\mathrm{op}} := \{g^{\mathrm{op}} \mid g \in G_R\}$ for the set of all the mirror images of generators. We define a prop

$$\mathbf{SFG}_R^+ := \mathbf{Free}\left(G_R \sqcup G_R^{\mathrm{op}}\right). \tag{5.81}$$

We call a morphism in the prop $\mathbf{SFG}_R^+$ a *(non-simplified) signal flow graph*: these extend our simplified signal flow graphs from Definition 5.45 because now we can also use the mirrored icons. By the universal property of free props, since we have said what the behavior of the generators is (the behavior of a reversed icon is the transposed relation; see Eq. (5.76)), we have specified the behavior of any signal flow graph.

The following two exercises help us understand what this behavior is.

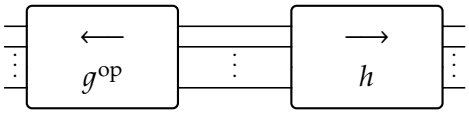
Exercise 5.82. Let $g\colon m \to n$, $h\colon \ell \to n$ be signal flow graphs. Note that $h^{\mathrm{op}}\colon n \to \ell$ is a signal flow graph, and we can form the composite $g \fatsemi (h^{\mathrm{op}})$:



Show that the behavior of $g \fatsemi (h^{\mathrm{op}}) \subseteq R^m \times R^\ell$ is equal to

$$\mathsf{B}(g \fatsemi (h^{\mathrm{op}})) = \{(x, y) \mid S(g)(x) = S(h)(y)\}. \qquad \lozenge$$

Exercise 5.83. Let $g\colon m \to n$, $h\colon m \to p$ be signal flow graphs. Note that $(g^{\mathrm{op}})\colon n \to m$ is a signal flow graph, and we can form the composite $g^{\mathrm{op}} \fatsemi h$



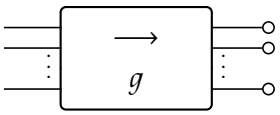
Show that the behavior of $g^{\mathrm{op}} \fatsemi h$ is equal to

$$\mathsf{B}((g^{\mathrm{op}}) \fatsemi h) = \{(S(g)(x), S(h)(x)) \mid x \in R^m\}. \qquad \lozenge$$

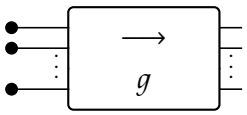
Linear algebra via signal flow graphs. In Eq. (5.75) we see that every matrix, or linear map, can be represented as the behavior of a signal flow graph, and in Exercise 5.82 we see that solution sets of linear equations can also be represented. This includes central concepts in linear algebra, like kernels and images.

Exercise 5.84. Here is an exercise for those that know linear algebra, in particular kernels and cokernels. Let R be a field, let $g\colon m \to n$ be a signal flow graph, and let $S(g) \in \mathbf{Mat}(R)$ be the associated $(m \times n)$-matrix (see Theorem 5.53).

1. Show that the composite of g with 0-reverses, shown here



   is equal to the kernel of the matrix $S(g)$.
2. Show that the composite of discard-reverses with g, shown here



   is equal to the image of the matrix $S(g)$.
3. Show that for any signal flow graph g, the subset $\mathsf{B}(g) \subseteq R^m \times R^n$ is a linear subspace. That is, if $b_1, b_2 \in \mathsf{B}(g)$ then so are $b_1 + b_2$ and $r * b_1$, for any $r \in R$. $\lozenge$

We have thus seen that signal flow graphs provide a uniform, compositional language to talk about many concepts in linear algebra. Moreover, in Exercise 5.84 we showed that the behavior of a signal flow graph is a linear relation, i.e. a relation whose elements can be added and multiplied by scalars $r \in R$. In fact the converse is true too: any linear relation $B \subseteq R^m \times R^n$ can be represented by a signal flow graph.

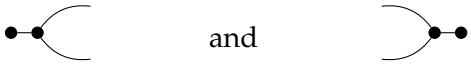
Exercise 5.85. One might want to show that linear relations on R form a prop, $\mathbf{LinRel}_R$. That is, one might want to show that there is a sub-prop of the prop $\mathbf{Rel}_R$ from Definition 5.79, where the morphisms $m \to n$ are the subsets $B \subseteq R^m \times R^n$ such that B is linear. In other words, where for any $(x, y) \in B$ and $r \in R$, the element $(r * x, r * y) \in R^m \times R^n$ is in B, and for any $(x', y') \in B$, the element $(x + x', y + y')$ is in B.

This is certainly doable, but for this exercise, we only ask that you prove that the composite of two linear relations is linear. ◊

Just like we gave a sound and complete presentation for the prop of matrices in Theorem 5.60, it is possible to give a sound and complete presentation for linear relations on R. Moreover, it is possible to give such a presentation whose generating set is $G_R \sqcup G_R^{\mathrm{op}}$ as in Eq. (5.81) and whose equations include those from Theorem 5.60, plus a few more. This presentation gives a graphical method for doing linear algebra: an equation between linear subspaces is true *if and only if* it can be proved using the equations from the presentation.

Although not difficult, we leave the full presentation to further reading (Section 5.5). Instead, we'll conclude our exploration of the prop of linear relations by noting that some of these 'few more' equations state that relations—just like co-design problems in Chapter 4—form a compact closed category.

Compact closed structure. Using the icons available to us for signal flow graphs, we can build morphisms that look like the 'cup' and 'cap' from Definition 4.58:



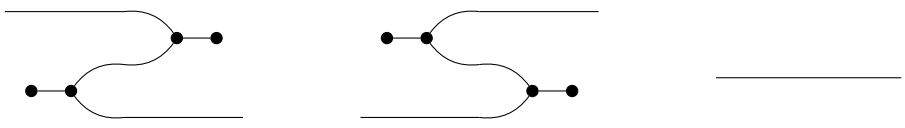
(5.86)

The behaviors of these graphs are respectively

$$\{(0, (x, x)) \mid x \in R\} \subseteq R^0 \times R^2 \quad \text{and} \quad \{((x, x), 0) \mid x \in R\} \subseteq R^2 \times R^0.$$

In fact, these show the object 1 in the prop $\mathbf{Rel}_R$ is dual to itself: the morphisms from Eq. (5.86) serve as the η_1 and ϵ_1 from Definition 4.58. Using monoidal products of these morphisms, one can show that any object in $\mathbf{Rel}_R$ is dual to itself.

Graphically, this means that the three signal flow graphs

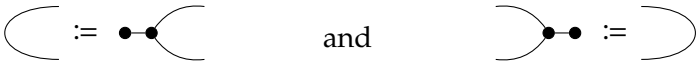


all represent the same relation.

Using these relations, it is straightforward to check the following result.

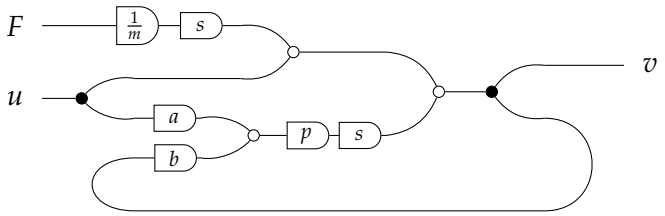
Theorem 5.87. The prop $\mathbf{Rel}_R$ is a compact closed category in which every object $n \in \mathbb{N}$ is dual to itself, $n = n^*$.

To make our signal flow graphs simpler, we define new icons cup and cap by the equations



Back to control theory. Let's close by thinking about how to represent a simple control theory problem in this setting. Suppose we want to design a system to maintain the speed of a car at a desired speed u. We'll work in signal flow diagrams over the rig $\mathbb{R}[s, s^{-1}]$ of polynomials in s and s^{-1} with coefficients in $\mathbb{R}$ and where $ss^{-1} = s^{-1}s = 1$. This is standard in control theory: we think of s as integration, and s^{-1} as differentiation.

There are three factors that contribute to the actual speed v. First, there is the actual speed v. Second, there are external forces F. Third, we have our control system: this will take some linear combination $a * u + b * v$ of the desired speed and actual speed, amplify it by some factor p to give a (possibly negative) acceleration. We can represent this system as follows, where m is the mass of the car.



This can be read as the following equation, where one notes that v occurs twice:

$$v = \int \frac{1}{m} F(t)dt + u(t) + p \int au(t) + bv(t)dt.$$

Our control problem then asks: how do we choose a and b to make the behavior of this signal flow graph close to the relation $\{(F, u, v) \mid u = v\}$? By phrasing problems in this way, we can use extensions of the logic we have discussed above to reason about such complex, real-world problems.

5.5 Summary and further reading

The goal of this chapter was to explain how props formalize signal flow graphs, and provide a new perspective on linear algebra. To do this, we examined the idea of free and presented structures in terms of universal properties. This allowed us to build props that exactly suited our needs.

Paweł Sobociński's *Graphical Linear Algebra* blog is an accessible and fun exploration of the key themes of this chapter, which goes on to describe how concepts such as determinants, eigenvectors, and division by zero can be expressed using signal flow

graphs [Sob]. For the technical details, one could start with Baez and Erbele [BE15], or Zanasi's thesis [Zan15] and its related series of papers [BSZ14; BSZ15; BS17]. For details about applications to control theory, see [FSR16]. From the control theoretic perspective, the ideas and philosophy of this chapter are heavily influenced by Willems' behavioral approach [Wil07].

For the reader that has not studied abstract algebra, we mention that rings, monoids, and matrices are standard fare in abstract algebra, and can be found in any standard introduction, such as [Fra67]. Rigs, also known as semirings, are a bit less well known, but no less interesting; a comprehensive survey of the literature can be found in [Gla13].

Perhaps the most significant idea in this chapter is the separation of structure into syntax and semantics, related by a functor. This is not only present in the running theme of studying signal flow graphs, but in our aside Section 5.4.2, where we talk, for example, about monoid objects in monoidal categories. The idea of functorial semantics is yet another due to Lawvere, first appearing in his thesis [Law04].

18.S097 Applied Category Theory

January IAP 2019