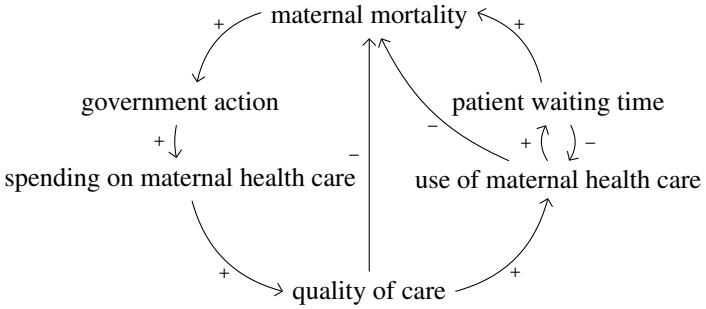
GRAPHS WITH POLARITIES

JOHN C. BAEZ[1,2] AND ADITTYA CHAUDHURI[3,4]

Abstract. In fields ranging from business to systems biology, directed graphs with edges labeled by signs are used to model systems in a simple way: the nodes represent entities of some sort, and an edge indicates that one entity directly affects another either positively or negatively. Multiplying the signs along a directed path of edges lets us determine indirect positive or negative effects, and if the path is a loop we call this a positive or negative feedback loop. Here we generalize this to graphs with edges labeled by a monoid, whose elements represent 'polarities' possibly more general than simply 'positive' or 'negative'. We study three notions of morphism between graphs with labeled edges, each with its own distinctive application: to refine a simple graph into a complicated one, to transform a complicated graph into a simple one, and to find recurring patterns called 'motifs'. We construct three corresponding symmetric monoidal double categories of 'open' graphs. We study feedback loops using a generalization of the homology of a graph to homology with coefficients in a commutative monoid. In particular, we describe the emergence of new feedback loops when we compose open graphs using a variant of the Mayer–Vietoris exact sequence for homology with coefficients in a commutative monoid.

1. Introduction

Graphs with edges labeled by elements of a fixed set are widely used as qualitative models of systems. For example:



This is a model of how health care can affect maternal mortality, a tiny simplified fragment of some larger models in the literature [31, 43]. Vertices of this graph represent various variables in a system. Labeled edges indicate how one variable can directly affect another. The label set in this example is $\{+, -\}$. An edge labeled by $+$ indicates that one variable has a direct positive effect on another: that is, increasing the former tends to increase the latter, everything else being equal. Similarly, an edge labeled by $-$ indicates that one variable has a direct negative effect on another. In fact the set $\{+, -\}$ naturally has the structure of a monoid, with this multiplication table:

$\cdot$	$+$	$-$
$+$	$+$	$-$
$-$	$-$	$+$

Other monoids are also commonly used to label edges in diagrams of this sort, and even more could be useful; we discuss these alternatives throughout the paper. The elements of such labeling sets are sometimes called 'polarities'.

[1] School of Mathematics, University of Edinburgh, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, UK EH9 3FD

[2] Department of Mathematics, University of California, Riverside CA, USA 92521

[3] Institute of Computer Science, University of Rostock, Albert-Einstein-Str. 22, 18059 Rostock, Germany

[4] Statistics and Mathematics Unit, Indian Statistical Institute Kolkata, 203 Barrackpore Trunk Road, Kolkata 700108, India

E-mail address: baez@math.ucr.edu, chaudhuriaditya@gmail.com.

Graphs with edges labeled by elements of $\{+, -\}$ are called 'causal loop diagrams' in the modeling tradition known as 'system dynamics'. System dynamics was first thoroughly explained in Forrester's 1961 book *Industrial Dynamics*, with the main application being to model fluctuations in supply chains [18]. Later Forrester collaborated with the mayor of Boston and applied the ideas to urban planning in his book *Urban Dynamics* [19]. In 2000, Sterman wrote an influential text *Business Dynamics* applying these ideas to the internal functioning of a business [45]. Later, system dynamics became widely used in health care and other subjects, and in 2014, Hovmand emphasized the process of building models by gathering and synthesizing information from diverse stakeholders in *Community Based System Dynamics* [23]. By now there is a thriving community of systems dynamics practitioners, with an annual international conerence, and software that helps them work with causal loop diagrams.

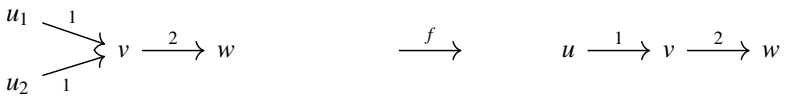
In a parallel line of development, biologists began using graphs with labeled edges to indicate how one type of biomolecule can promote or inhibit the production of another. When applied to the expression of genes, these graphs are called 'gene regulatory networks' [15, 25]. These and other networks became important in 'systems biology', a holistic approach to biological systems that aims to understand how the interactions between their components lead to emergent behaviors [3, 27]. Since the 1990s, large databases of biological networks have been created, integrating visualizations with molecular data, ontologies and references to the literature [24]. Eventually standardization became necessary, and a committee was formed to develop Systems Biology Graphical Notation (SBGN) [30]. This is a system of visual representations of biological networks to facilitate clear, consistent communication and the reuse of information. Over time, SBGN has become a widely adopted standard for visualizing biological networks at varying levels of detail. It encompasses three distinct yet complementary visual languages:

- *Process Description* (SBGN-PD) focuses on detailed biochemical reactions [42].
- *Activity Flow* (SBGN-AF) illustrates the flow through which various biochemical entities and activities influence each other [38].
- *Entity Relationship* (SBGN-ER) highlights how entities affect each other's actions and behaviors [44].

The second author and his collaborators have recently attempted to develop a compositional framework for SBGN-PD diagrams via the theory of 'open process networks' [13], motivated by the theory of open Petri nets [8]. The current work is instead connected to SBGN-AF diagrams.

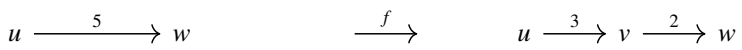
We study three kinds of morphisms between labeled graphs, which require increasing amounts of structure on the label set:

(1) 'maps' between L-labeled graphs for a set L. Here is an example taking $L = \mathbb{N}$:



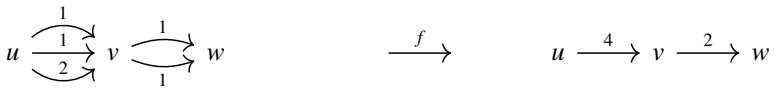
Here the edge labels simply get pulled back along a map of graphs. As explained in Section 2, maps between set-labeled graphs can be useful for refining a simple model of a system to a more complex model. The more complex model is the *source* of the map of L-labeled graphs; in the example above we refined a model with one entity called u to one with two similar entities called u_1 and u_2.

(2) 'Kleisli morphisms' between M-labeled graphs for a monoid M. Here is an example taking $M = \mathbb{N}$ made into a monoid using addition:



Here an edge can get mapped to a path of edges if its label is the sum of their labels. In Section 3 we argue that many important sets of labels in systems dynamics and systems biology are in fact monoids. Section 4 explains how Kleisli morphisms can be used to find 'motifs': simple patterns commonly which play important structural roles in many systems.

(3) 'additive morphisms' between finite C-labeled graphs whenever C is a commutative monoid. Here is an example taking $C = \mathbb{N}$ made into a commutative monoid using addition:



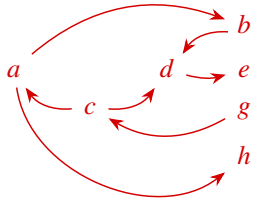
Here when several edges get mapped to the same edge, we sum their labels. In systems dynamics and systems biology, most of the monoids of labels are commutative. Section 5 explains how additive morphisms can be used to simplify a complex model of a system.

We also study 'open' labeled graphs, which are roughly labeled graphs with some vertices specified as 'inputs' or 'outputs'. The advantage of these is that large labeled graphs can be build up out of small open labeled graphs by a process of composition. We consider three variants of this theme:

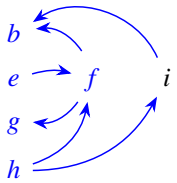
(1) In Theorem 7.2, for any set L we construct a symmetric monoidal double category of open L-labeled graphs and maps between these. This uses the theory of structured cospans [6, 7, 14].
(2) In Theorem 7.6, for any monoid M we construct a symmetric monoidal double category of open M-labeled graphs and Kleisli morphisms between these. This pushes the theory of structured cospans slightly beyond its usually stated level of generality.
(3) In Theorem 7.7, for any commutative monoid C we construct a symmetric monoidal double category of open C-labeled finite graphs and additive morphisms between these. This seems to require the theory of decorated cospans [7, 16, 17].

We also study feedback loops in graphs, and how new feedback loops can emerge when one composes open graphs. To study these, in Section 8 we define the first homology monoid of a graph G with coefficients in a commutative monoid C. When C is an abelian group this monoid reduces to the familiar homology group with coefficients in C, which does not depend on the direction of the edges of G. When C is a monoid without inverses, such as $\mathbb{N}$ with addition, the first homology detects *directed* loops in G. This is crucial for the study of feedback, since an edge $A \to B$ indicates that A affects B, but not necessarily vice versa. We carry out a detailed study of the first homology of G with coefficients in $\mathbb{N}$, showing in Theorem 8.12 that it is generated by homology classes of 'simple loops': that is, directed loops that do not cross themselves.

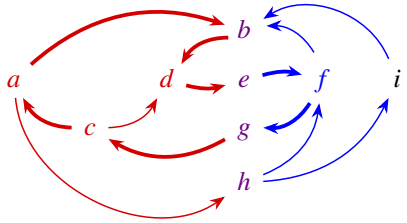
In Section 9 we study how directed loops emerge when we compose open graphs. For example, this graph has no directed loops:



but when we glue it to the following graph, which also has no directed loops:



by identifying the vertices called b, e, g and h, we obtain a graph with directed loops, one shown in boldface:



In the case of undirected loops the phenomenon of 'emergent loops' is well understood by the van Kampen theorem. For homology with coefficients in an abelian group the analogous phenomenon is well understood using the Mayer–Vietoris exact sequence [22]. We prove some related theorems for directed loops and homology with coefficients in $\mathbb{N}$.

Acknowledgements. We thank Nathaniel Osgood for teaching us about causal loop diagrams and system dynamics, and Olaf Wolkenhauer for suggesting the application of causal loop diagrams to Systems Biology Graphical Notation – Activity Flow (SBGN-AF). We thank Ralf Köhl, Olaf Wolkenhauer, Clémence Réda, Shailendra Gupta, and Rahul Bordoloi for valuable discussions on the applications of causal loop diagrams to SBGN-AF. We thank Oscar Cunningham, Morgan Rogers and Zoltan Kocsis for dispelling our hope that the first homology monoid of a directed graph is always free, and Tobias Fritz for explaining a generalization of homology for commutative monoids. We thank David Egolf for suggesting the idea of a monoid containing an element indicating that one vertex has a direct effect on another that we are deciding to neglect. We thank Evan Patterson for discussions involving Catcolab's treatment of these ideas, and Kevin Carlson and others for general help with category theory.

Notation. In this paper we use a sans-serif font like C for categories, boldface like $\mathbf{B}$ for bicategories or 2-categories, and blackboard bold like $\mathbb{D}$ for double categories. Thus, Cat denotes the category of small categories while **Cat** denotes the 2-category of small categories. For double categories with names having more than one letter, like $\mathbb{C}\mathbf{sp}(\mathsf{X})$, only the first letter is in blackboard bold.

2. Set-labeled graphs

Although a graph captures the interconnection between a set of elements, it does not say anything about the *types of interconnection*. We can do this using labeled graphs with edges labeled by elements of a set.

First, some preliminaries. For us, a **graph** $G = (E, V, s, t)$ consists of a set of **vertices** V and a set of **edges** E together with a **source map** $s\colon E \to V$ and a **target map** $t\colon E \to V$. We define a **map of graphs** $f = (f_0, f_1)$ from a graph $G = (E, V, s, t)$ to a graph $G' = (E', V', s', t)$ to be a pair of functions $f_0\colon V \to V'$ and $f_1\colon E \to E'$ such that the following two diagrams commute:

$$\begin{array}{ccc} E & \xrightarrow{f_1} & E' \\ {\scriptstyle s}\downarrow & & \downarrow{\scriptstyle s'} \\ V & \xrightarrow[f_0']{} & V' \end{array} \qquad \begin{array}{ccc} E & \xrightarrow{f_1} & E' \\ {\scriptstyle t}\downarrow & & \downarrow{\scriptstyle t'} \\ V & \xrightarrow[f_0']{} & V'. \end{array}$$

The collection of graphs and maps between them form a category, which we denote by Gph. The category Gph is equivalent to the functor category $\mathsf{Set}^{\mathsf{G}}$, where G is the category freely generated by two parallel morphisms $e \rightrightarrows v$. Since $\mathsf{G} \cong \mathsf{G}^{\mathrm{op}}$ we can also say Gph is equivalent to $\mathsf{Set}^{\mathsf{G}^{\mathrm{op}}}$. Thus, Gph is a presheaf topos and it enjoys all the privileges of such categories: it is complete (has all limits) and cocomplete (has all colimits), cartesian closed, has a subobject classifier, etc. When we replace the category Set with the category of finite sets FinSet, we obtain a category

$$\mathsf{FinGph} \simeq \mathsf{FinSet}^{\mathsf{G}}$$

called the category of **finite graphs**, whose objects are graphs with finitely many vertices and edges. This category is still a topos, so it has finite limits, finite colimits, it is cartesian closed, and has a subobject classifier.

Now we turn to the simplest sort of labeled graph: a graph whose edges are labeled by some fixed set.

Definition 2.1. Given a set L, we define a ***L*-labeled graph** (G, ℓ) to be a graph $G = (E, V, s, t)$ equipped with a **labeling map** $\ell\colon E \to L$. We call L the set of **labels**.

A map of labeled graphs is a map between their underlying graphs that preserves the labels of edges:

Definition 2.2. Given L-labeled graphs (G, ℓ) and (G', ℓ'), define a **map of *L*-labeled graphs** to be a map of graphs $f\colon G \to G'$ that makes the following triangle commute:

$$\begin{array}{ccc} E & \xrightarrow{f_1} & E' \\ & {\scriptstyle \ell}\searrow \quad \swarrow{\scriptstyle \ell'} & \\ & L & \end{array}$$

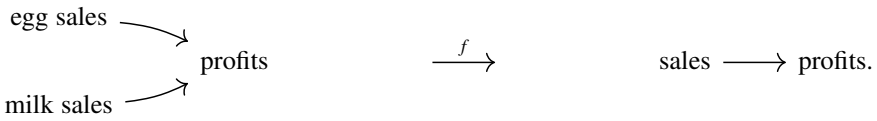
We can pull back an L-labeling along a map of graphs:

Lemma 2.3. *Let L be a set and (G, ℓ) an L-labeled graph. For any map of graphs $f\colon G' \to G$, there is a unique L-labeling of G', called the* **pullback of ℓ along f** *and denoted $f^*\ell$, such that f is a map of L-labeled graphs from $(G', f^*\ell)$ to (G, ℓ). The pullback is contravariantly functorial in the following sense:*

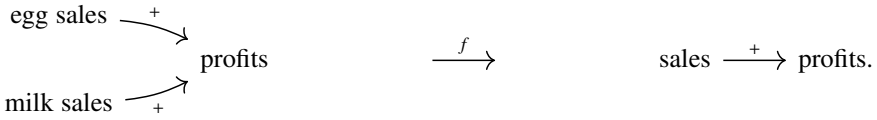
$$(f \circ g)^* = g^* \circ f^*, \qquad 1_* = 1.$$

Proof. We are forced to take $f^*\ell = \ell \circ f_1$, and this works. □

To see how this fact can be used in system dynamics or systems biology, consider this map of graphs:



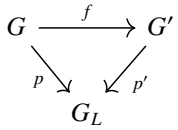
We can think of the first graph as a more complicated model in which one vertex in the second model has been differentiated into two. Now suppose we label the edges of the second graph by elements of $L = \{+, -\}$. This graph has just one edge, and if increased sales leads to increased profits (all else being equal) we label this edge with $+$. Then Lemma 2.3 says there is a unique way to label the edges of the first graph that lets us lift the map f to a map of L-labeled graphs:



The functoriality in Lemma 2.3 says that one can start with a model of a system as an L-labeled graph and successively refine it by choosing graphs that map to the original graph, with the L-labelings being automatically inherited all these more refined graphs in a consistent way. In system dynamics this and related processes of model refinement are called 'stratification' [9].

In mathematical terms, there is a forgetful functor sending L-labeled graphs to their underlying graphs, and Lemma 2.3 says this functor is a 'discrete fibration'. To better understand this and other facts about labeled graphs it can be useful to take a more sophisticated outlook. The reader less familiar with category theory can skip the rest of this section. For an introduction to discrete fibrations see [32, Sec. 2.1].

First, note that a L-labeled graph can be seen as a graph over G_L, the graph with only one vertex and an edge for each element $\ell \in L$. That is, an L-labeled graph is the same as graph G equipped with a map of graphs $p\colon G \to G_L$. Similarly, a map of L-labeled graphs can be seen as a map of graphs over L, i.e., a map of graphs $f\colon G \to G'$ such that this triangle commutes:



Thus, the category of L-labeled graphs and maps between them is isomorphic to the slice category Gph/G_L.

This makes it easy to deduce the following facts from general principles:

Proposition 2.4. *For any set L, we have the following:*

(a) The category Gph/G_L is a presheaf topos.
(b) The forgetful functor $U\colon \mathsf{Gph}/G_L \to \mathsf{Gph}$ is a discrete fibration.
(c) The presheaf $\mathcal{P}\colon \mathsf{Gph}^{\mathrm{op}} \to \mathsf{Set}$ associated to U is representable by G_L.
(d) The category Gph/G_L is equivalent to the category of elements $\int \mathcal{P}$.

Proof. These follow from more general results. If we write $\widehat{\mathsf{C}}$ for the category of presheaves on a category C, and choose a presheaf $F \in \widehat{C}$, then:

(a) The category $\widehat{\mathsf{C}}/F$ is a presheaf topos.
(b) The forgetful functor $U\colon \widehat{\mathsf{C}}/F \to \widehat{\mathsf{C}}$ is a discrete fibration.
(c) The functor $\mathcal{P}\colon \widehat{\mathsf{C}}^{\mathrm{op}} \to \mathsf{Set}$ associated to U is representable by F.
(d) The category $\widehat{\mathsf{C}}/F$ is equivalent to the category of elements $\int \mathcal{P}$.

The proposition follows taking $\mathsf{C} = \mathsf{G}$ and $F = G_L$. These more general results are well-known, but here we sketch them, with references:

(a) follows from the fact $\widehat{\mathsf{C}}/F \simeq \widehat{\int F}$, where $\int F$ is the category of elements of F [33, Exercise III.8]. In our example of interest, one can check directly that the category of L-labeled graphs is equivalent to the category of presheaves on a category with one object v, one object for each edge label $\ell \in L$, and two morphisms $s_\ell, t_\ell \colon v \to \ell$ for each ℓ.

(b) For any object d in a category D, to say that the forgetful functor $U \colon \mathsf{D}/d \to D$ is a discrete fibration means that given any object over d, say $p \colon e \to d$, and any morphism $f \colon e' \to e$ in D, there exists a unique morphism g in D/d such that $U(g) = f$. This is easily checked taking g as follows:

$$\begin{array}{ccc} e' & \xrightarrow{f} & e \\ & {\scriptstyle p\circ f}\searrow \quad \swarrow{\scriptstyle p} & \\ & d. & \end{array}$$

(c) This holds because the fiber of U at any $G \in \widehat{C}$ is of the form $\hom(G, F)$.

(d) From Grothendieck's correspondence between discrete fibrations and presheaves [32, Sec. 2.1] we have the following isomorphism of discrete fibrations

$$\begin{array}{ccc} \int \mathcal{P} & \xrightarrow{\cong} & \widehat{\mathsf{C}}/F \\ & \searrow \quad \swarrow & \\ & \widehat{\mathsf{C}} & \end{array}$$

and thus the category $\int \mathcal{P}$ is equivalent to $\widehat{\mathsf{C}}/F$. □

The discrete fibration in Proposition 2.4 (b) implies that there is a functorial way to pull back L-labelings along maps of graphs, as we have already seen. We can also push forward labelings along a map between labeling sets. That is, given a map $\phi \colon L \to L'$, and an L-labeled graph (G, ℓ), we can define an L'-labeled graph

$$\phi_*(G, \ell) := (G, \phi \circ \ell).$$

This process is functorial:

$$(\phi \circ \psi)_* = \phi_* \circ \psi_*, \qquad 1_* = 1.$$

More formally, we have the following result.

Proposition 2.5. *The following assignment*

$$\begin{array}{rccc} F \colon & \mathsf{Set} & \to & \mathsf{Cat} \\ \\ & L & \mapsto & \mathsf{Gph}/G_L \\ & (L \xrightarrow{\phi} L') & \mapsto & (\mathsf{Gph}/G_L \xrightarrow{\phi_*} \mathsf{Gph}/G_{L'}) \end{array}$$

defines a functor.

Proposition 2.5 lets us define a single **category of set-labeled graphs** that allows for all possible choices of label set L. This is simply the category of elements $\int F$ of the functor F. Explicitly:

Proposition 2.6. *$\int F$ is the category where:*

- *an object is a pair $(L, p \colon G \to G_L)$,*
- *a morphism from $(L, p \colon G \to G_L)$ to $(L', p' \colon G' \to G'_L)$ is a pair (ϕ, f), where $\phi \colon L \to L'$ is a function and $f \colon G \to G'$ is a map of graphs such that this square commutes:*

$$\begin{array}{ccc} G & \xrightarrow{f} & G' \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p'} \\ G_L & \xrightarrow[\tilde{\phi}]{} & G_{L'}. \end{array}$$

where $\tilde{\phi} \colon G_L \to G_{L'}$ maps each edge $\ell \in L$ of G_L to the edge $\phi(\ell) \in L'$ of $G_{L'}$.

3. Monoid-labeled graphs

We now turn to graphs where the edges are labeled by elements of a monoid. We think of these monoid elements as 'polarities': ways for the entity corresponding to one vertex to affect another. If a graph has an edge from a vertex u to a vertex v labeled with a monoid element m, and an edge from v to a vertex w labeled with a monoid element m', we say that u has an indirect effect on w equal to $m'm$.

Here are some monoids that are useful for describing polarities:

Example 3.1. The terminal monoid is a monoid containing just one element. This is also known as the trivial group. In the applications at hand we write the group operation as multiplication and call the one element 1, so that $1 \cdot 1 = 1$. Thus, we call this monoid $\{1\}$. Any graph becomes a $\{1\}$-graph in a unique way, by labeling each edge with 1, and this gives an isomorphism of categories

$$\{1\}\mathsf{Gph} \cong \mathsf{Gph}.$$

We can use a graph to describe causality in at least two distinct ways. Suppose v and w are vertices of a graph.

(1) We can use the presence of an edge from v to w to indicate that the entity named by v has a *direct effect* on the entity named by w, and the absence of an edge to indicate that v has *no direct effect* on w. (Even if there is no edge from v to w, v may still have an indirect effect on w if there is path of edges from v to w.)

(2) We can use the presence of an edge from v to w to indicate that v has a *direct effect* on w, and the absence of an edge to indicate that v has *has no currently known direct effect* on w. This interpretation is useful for situations where we start with a graph having no edges, and add an edge each time we discover that one vertex has a direct effect on another.

We can also take at least three different attitudes to the presence of multiple edges from one vertex to another:

(a) We can treat them as redundant, hence unnecessary, allowing us to simplify any graph so that it has at most one edge from one vertex to another.

(b) We can treat them as indicating different ways in which one vertex directly affects another.

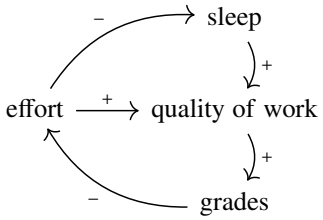
(c) We can use the number of edges from v to w to indicate the amount by which v affects w.

All these subtleties of interpretation can also arise for M-graphs where M is any other monoid. We will not mention them each time, but in applications it can be important to clearly fix an interpretation.

Example 3.2. The most widely used monoid in this field is the abelian group $\mathbb{Z}/2$. It is typical to write the group operation as multiplication rather than addition, and call this group $\{+, -\}$:

$\cdot$	$+$	$-$
$+$	$+$	$-$
$-$	$-$	$+$

In the field called System Dynamics [46, Chap. 5], a graph with edges labeled by elements of $\{+, -\}$ is called a 'causal loop diagram'. Here is a causal loop diagram serving as a simple model of students doing homework:



The more effort a student puts into homework, the less sleep they get, so the edge from 'effort' to 'sleep' is labeled with a $-$, and so on.

In biology, the simplest sort of regulatory network [2] is a graph with edges labeled by elements of $\{+, -\}$. However, the notation is different. Instead of an edge labeled with $+$, an arrow $A \longrightarrow\!\!\triangleright\, B$ indicates that A 'promotes' or 'stimulates' B, while an arrow $A \longrightarrow\!\!\dashv\, B$ indicates that A 'represses' or 'inhibits' B. In the official SBGN-AF terminology, $A \longrightarrow\!\!\triangleright\, B$ means that A is a 'positive influence' on B, while $A \longrightarrow\!\!\dashv\, B$ means that A has a 'negative influence' on B.

Example 3.3. Another important monoid of polarities is the multiplicative monoid of $\mathbb{Z}/3$, which we can write either as $\{1, 0, -1\}$ or $\{+, 0, -\}$. Now the multiplication table is this:

$\cdot$	$+$	0	$-$
$+$	$+$	0	$-$
0	0	0	0
$-$	$-$	0	$+$

This monoid contains the one in the previous example as a submonoid, but also a new element 0 that is **absorbing**: $0 \cdot x = x \cdot 0 = 0$ for all x. So, unlike $\{+, -\}$, this monoid $\{+, 0, -\}$ is not a group.

There are at least three distinct interpretations of $\{+, 0, -\}$-graphs:

(1) We can use + to denote a positive effect, − to denote a negative effect, and 0 to denote *an effect whose value varies depending on the circumstances*.
(2) We can use + to denote a positive effect, − to denote a negative effect, and 0 to denote *an effect that we deem negligible*.
(3) We can use + to denote a positive effect, − to denote a negative effect, and 0 to denote *an unknown effect*.

Our choice of interpretation affects how we interpret the absence of an edge from one vertex to another. We discuss this in Example 3.4.

We believe that in SBGN-AF diagrams, this new element 0 is used to mean an 'unknown influence', and and denoted with a diamond-headed arrow. Thus, $A \longrightarrow\!\!\diamond\, B$ means that A has an unknown influence on B.

Example 3.4. More generally, for any monoid M we can form a new monoid $M \cup \{0\}$ where 0 is a new absorbing element. Thus, in $M \cup \{0\}$ the product of elements of M is defined as before, but $m \cdot 0 = 0 \cdot m = 0$ for all $m \in M$, and $0 \cdot 0 = 0$. This new monoid contains M as a submonoid. As before, there are at least three interpretations of the element 0:

(1) We can use 0 to denote an effect whose value varies depending on the circumstances. In this case we can use the absence of an edge from a vertex v to a vertex w to indicate either:
   (a) there is *no direct effect* of v on w, or
   (b) the effect of v on w is *unknown*.
(2) We can use 0 to denote an effect that we deem negligible. In this case we can again use either interpretation (a) or (b) of the absence of an edge.
(3) We can use 0 to denote an unknown effect. In this case we can use interpretation (a) of the absence of an edge, but interpretation (b) is awkward, because then the absence of an edge means the same thing as an edge labeled by 0.

Example 3.5. For any monoid M we can also form a new monoid $M \cup \{I\}$ where I is a new identity element. Thus, in $M \cup \{I\}$ the product of M is defined as before, but $m \cdot I = I \cdot m = m$ for all $m \in M$, and $I \cdot I = I$. When we apply this construction to $\{+, -\}$ we obtain a monoid $\{+, I, -\}$ with the following multiplication table:

$\cdot$	I	$+$	$-$
I	I	$+$	$-$
$+$	$+$	$+$	$-$
$-$	$-$	$-$	$+$

We believe that in SBGN-AF, this new element I has the meaning of a 'necessary stimulation', and it is denoted with a barred arrow, so $A \longrightarrow\!\!\triangleright\, B$ means that A is *necessary* for B.

If we adjoin an absorbing element to this monoid $\{I, +, -\}$, we obtain a monoid $\{I, +, 0, -\}$ that can handle a large fraction of SBGN-AF diagrams. But in these diagrams:

- instead of an edge labeled I, we write $\longrightarrow\!\!\triangleright$ for 'necessary stimulation'.
- instead of an edge labeled +, we write $\longrightarrow\!\triangleright$ for 'positive influence'.
- Instead of an edge labeled 0, we write $\longrightarrow\!\diamond$ for 'unknown influence'.
- Instead of an edge labeled −, we write $\longrightarrow\!\!\dashv$ for 'negative influence'.

We can also obtain this monoid $\{I, +, 0, -\}$ by adjoining a new identity element to the monoid $\{+, 0, -\}$ discussed in Example 3.3. Either way, the multiplication table is as follows:

$\cdot$	I	$+$	0	$-$
I	I	$+$	0	$-$
$+$	$+$	$+$	0	$-$
0	0	0	0	0
$-$	$-$	$-$	0	$+$

Example 3.6. The multiplicative group of the reals, $(\mathbb{R} - \{0\}, \cdot, 1)$, consists of all real numbers except zero, with multiplication as its monoid operation. The monoid $\{+, -\}$ of Example 3.2 gives purely qualitative information about whether an effect is positive or negative. We can label the edges of a graph with elements of $\mathbb{R} - \{0\}$ to give quantitative information about *how much* of a positive or negative direct effect one vertex has on another.

Example 3.7. The multiplicative monoid of the reals, $(\mathbb{R}, \cdot, 1)$, consists of all real numbers with multiplication as its monoid operation. This is obtained from the multiplicative group of the reals by adjoining an absorbing element as in Example 3.4.

Example 3.8. The monoid $([0, \infty), +, 0)$ consists of nonnegataive real numbers with addition as its monoid operation. We can use this monoid to describe delayed effects: an edge labeled with a real number $t \geq 0$ can indicate that one vertex affects another after a delay of time t.

Example 3.9. The monoid $(\mathbb{N}, +, 0)$ consists of all natural numbers with addition as its monoid operation. We can use this monoid to describe delayed effects in discrete-time systems, using an edge labeled with a natural number n to indicate that one vertex affects another after a delay of n time steps.

Example 3.10. Products of monoids are also useful: for example, to describe both time delays and whether an effect is positive or negative, we can use the monoid $([0, \infty), +, 0) \times \{+, -\}$.

Example 3.11. All the above monoids above are commutative, and indeed commutative monoids are by far the most commonly used for polarities. We discuss special features of the commutative case in Sections 5 and 8. However, graphs with edges labeled by not-necessarily-commutative monoids do show up naturally in some contexts. For example, in computer science [20, Sec. 2.1], a **semiautomaton** consists of a set V of **states**, a set A of **inputs**, and a map $\alpha \colon A \to V^V$ that describes how each input acts on each state to give a new state. Let M be the monoid of maps from V to itself generated by all the maps $\alpha(a)$ for $a \in A$. Let G be the graph where:

- The set of vertices is V.
- The set of edges is $E = A \times V$.
- The source map is given by

$$\begin{array}{rccc} s\colon & E & \to & V \\ & (a, v) & \mapsto & v. \end{array}$$

- The target map is given by

$$\begin{array}{rccc} t\colon & E & \to & V \\ & (a, v) & \mapsto & \alpha(a)(v). \end{array}$$

  Since the monoid of maps $M \subseteq V^V$ is generated by elements $\alpha(a)$ for $a \in A$, there is an M-labeling of G given by

$$\ell(a, v) = \alpha(a).$$

  In short, a semi-automaton gives an monoid-labeled graph where the vertices represent states and for each input a mapping a state v to a state w there is an edge labeled by the monoid element $\alpha(a) \in M$. Note however that it also give an set-labeled graph with the same vertices, where for each input a mapping a state v to a state w there is an edge labeled by $a \in A$.

Given a monoid $(M, \cdot, 1)$, we define an M-labeled graph and a map of M-labeled graphs just as when M is a mere set (see Definitions 2.1 and 2.2). One advantage of using graphs with labelings $\ell \colon E \to M$ where M is a monoid is that for any path of edges in G

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{m-1}} v_{m-1} \xrightarrow{e_m} v_m$$

we can form the product $\ell(e_m)\cdots\ell(e_1)$, and use this to describe how the vertex v_0 *indirectly* affects the vertex v_n.

To formalize this, recall that there are adjoint functors

$$\mathsf{Gph} \underset{\text{Und}}{\overset{\text{Free}}{\rightleftarrows}} \mathsf{Cat}.$$

The functor $\mathrm{Und}\colon \mathsf{Cat} \to \mathsf{Gph}$ sends any category C to its underlying graph, whose vertices are objects of C and whose edges are morphisms. This has a left adjoint $\mathsf{Free}\colon \mathsf{Gph} \to \mathsf{Cat}$ sending each graph $G = (E, V, s, t)$ to the category $\mathsf{Free}(G)$ where:

- an object is a vertex $v \in V$.
- a morphism from v to w is a **path** in G:

  $$v = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{m-1}} v_{m-1} \xrightarrow{e_m} v_m = w$$

  where $m \geq 0$.
- composing paths

  $$v_0 \xrightarrow{e_1} \cdots \xrightarrow{e_m} v_m$$

  and

  $$v_m \xrightarrow{e_{m+1}} \cdots \xrightarrow{e_{m+n}} v_{m+n}$$

  gives the path

  $$v_0 \xrightarrow{e_1} \cdots \xrightarrow{e_m} v_m \xrightarrow{e_{m+1}} \cdots \xrightarrow{e_{m+n}} v_{m+n}$$
- the identity $1\colon v \to v$ is the path of length 0 starting and ending at v.

Suppose M is a monoid. Our next goal is to show that when (G, ℓ) is an M-labeled graph, $\mathsf{Free}(G)$ becomes a kind of M-labeled category—usually called an 'M-graded' category. Just as M-labeled graphs are conveniently treated as graphs over G_M, we can define an M-graded category to be a category C over $\mathsf{B}M$, the one-object category with one morphism for each element of M, with composition defined to be multiplication in M.

Definition 3.12. Let M be a monoid. The **category of M-graded categories** is the slice category $\mathsf{Cat}/\mathsf{B}M$.

Thus, an M-graded category $p\colon \mathsf{C} \to \mathsf{B}M$ is a category C for which each morphism is mapped to an element $p(f)$ of M called its **grade** in such a way that

$$p(fg) = p(f)p(g)$$

for any pair of composable morphisms f and g, and

$$p(1_c) = 1 \in M$$

for each $c \in \mathsf{C}$. A map of M-graded categories, say F from $p\colon \mathsf{C} \to \mathsf{B}M$ to $p'\colon \mathsf{C}' \to \mathsf{B}M'$, works out to be a functor $F\colon \mathsf{C} \to \mathsf{C}'$ with the property that

$$p'(F(f)) = p(f)$$

for every morphism in C. In other words, F preserves the grades of morphisms.

This category of monoid-graded categories enjoys certain nice properties analogous to the category of set-labeled graphs (see Proposition 2.4).

Proposition 3.13. *For any monoid M, we have the following:*

(a) The category $\mathsf{Cat}/\mathsf{B}M$ is locally finitely presentable, and thus complete and cocomplete.
(b) The forgetful functor $\bar{U}\colon \mathsf{Cat}/\mathsf{B}M \to \mathsf{Cat}$ is a discrete fibration.
(c) The presheaf $\bar{\mathcal{P}}\colon \mathsf{Cat}^{\mathrm{op}} \to \mathsf{Set}$ associated to $\mathcal{F}$ is representable by $\mathsf{B}M$.
(d) The category $\mathsf{Cat}/\mathsf{B}M$ is same as the category of elements $\int \bar{\mathcal{P}}$.

Proof. For (a), recall that a category is locally finitely presentable precisely when it is the category of models for a finite limits sketch [1]. In the case of $\mathsf{Cat}/\mathsf{B}M$, this follows because we can define a category over $\mathsf{B}M$ as having a set Ob of objects, for each $\ell \in M$ a set Mor_ℓ of ℓ-labeled morphisms, source and target maps $s_\ell, t_\ell\colon \mathrm{Mor}_\ell \to \mathrm{Ob}$, an identity-assigning map $\mathrm{id}\colon \mathrm{Ob} \to \mathrm{Mor}_1$, and composition maps defined on pullbacks

$$\circ_{\ell,\ell'}\colon \mathrm{Mor}_\ell \,{}_{s_\ell}\!\times_{t_\ell} \mathrm{Mor}_{\ell'} \to \mathrm{Mor}_{\ell'\ell}$$

satisfying associativity and the unit laws. Every locally finitely presentable category is complete and cocomplete [1].

Parts (b), (c) and (d) follow exactly the same way as in the proofs of the corresponding parts of Proposition 2.4. □

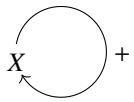
Natural transformations between maps of M-graded categories obey an interesting law. Suppose $(M, \cdot, 1)$ is a monoid, $p\colon \mathsf{C} \to \mathsf{B}M$, $p'\colon \mathsf{C}' \to \mathsf{B}M$ are a pair of M-graded categories, and $F, F'\colon \mathsf{C} \to \mathsf{C}'$ are a pair of maps of M-graded categories. Then, for any natural transformation $\eta\colon \mathrm{Ob}(\mathsf{C}) \to \mathrm{Mor}(\mathsf{C}')$ from F to F', we have

$$p(\gamma)p'(\eta(a)) = p'(\eta(b))p(\gamma)$$

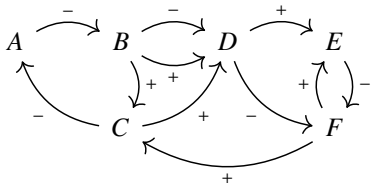
for all morphisms $f\colon a \to b$ in C.

4. Motifs in monoid-labeled graphs

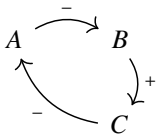
Small monoid-labeled graphs are often called 'motifs' because, like motifs in a piece of music, they show up repeatedly in a meaningful way in a larger context [4]. For example take $M = \{+, -\}$. Let (G, ℓ) be this M-labeled graph:



Let (H, m) be this larger M-labeled graph:



In a sense, there is a copy of (G, ℓ) hiding in (H, m). Here it is:



Here the entity A affects itself in a positive way, but *indirectly*, through B and C. There is not a map of M-labeled graphs from (G, ℓ) to (H, m). Instead, there is a functor from the free M-graded category on (G, ℓ) to the free M-graded category on (H, m), say

$$F\colon \mathsf{Free}_M(G, \ell) \to \mathsf{Free}_M(H, m).$$

This functor sends X to A and it sends the edge from X to itself, which is a morphism in $\mathsf{Free}_M(G, \ell)$, to the path from A to B to C to A, which is a morphism in $\mathsf{Free}_M(H, m)$. Moreover F is a map of M-graded categories. This is the sense in which the motif (G, ℓ) appears in the larger M-labeled graph (H, m).

To make this precise we need to understand the free M-graded category on an M-labeled graph.

Lemma 4.1. *Let M be a monoid. The forgetful functor sending M-labeled categories to their underlying M-labeled graphs,*

$$\mathrm{Und}_M\colon \mathsf{Cat}/\mathsf{B}M \to \mathsf{Gph}/G_M,$$

has a left adjoint

$$\mathsf{Free}_M\colon \mathsf{Gph}/G_M \to \mathsf{Cat}/\mathsf{B}M.$$

Proof. We define Free_M as follows. For any M-labeled graph (G, ℓ) we define $\mathsf{Free}_M(G, \ell)$ to have $\mathsf{Free}(G)$ as its underlying category, with the labeling of any morphism

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_n$$

in $\mathsf{Free}(G)$ being the product $\ell(e_n) \cdots \ell(e_1) \in M$. Recall that a map of L-labeled graphs $f\colon (G, \ell) \to (G', \ell')$ is a map f_0 sending vertices to vertices and a map f_1 sending edges to edges, preserving the labeling. Thus we define $\mathsf{Free}_M(f)$ to equal f_0 on objects of $\mathsf{Free}_M(G, \ell)$ and to send any morphism

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_n$$

to

$$f_0(v_0) \xrightarrow{f_1(e_1)} f_0(v_1) \xrightarrow{f_1(e_2)} \cdots \xrightarrow{f_1(e_{n-1})} f_0(v_{n-1}) \xrightarrow{f_1(e_n)} f_0(v_n).$$

One can check that Free_M is left adjoint to Und_M. □

Definition 4.2. For any monoid M, the **Kleisli category of M-labeled graphs**, $\mathsf{K}(\mathsf{Gph}/G_M)$ has:

- as objects, M-labeled graphs,
- as morphisms from an M-labeled graph (G, ℓ) to an M-labeled graph (G', ℓ'), maps of M-graded categories
$$f \colon \mathsf{Free}_M(G, \ell) \to \mathsf{Free}_M(G', \ell'),$$
- as composition, the usual composition of maps of M-graded categories,
- as identity morphism, the usual identity maps of M-graded categories.

We call a morphism in this category a **Kleisli morphism** between M-labeled graphs.

The concept of Kleisli category shows up whenever we have a pair of adjoint functors. Usually the Kleisli category is described in terms of a monad, which here would be the monad $\mathrm{Und}_M \circ \mathsf{Free}_M$. There is a bijection between morphisms

$$\mathsf{Free}_M(G, \ell) \to \mathsf{Free}_M(G', \ell')$$

and morphisms

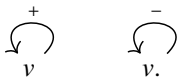
$$(G, \ell) \to (\mathrm{Und}_M \circ \mathsf{Free}_M)(G', \ell').$$

It is common to describe composition of morphisms of the former sort indirectly, in terms of morphisms of the latter sort. However, this is not needed for our purposes.

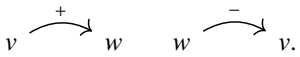
Aduddell et al. [2] introduced the Kleisli category of $\{+, -\}$-labeled graphs to study motifs in biochemical regulatory networks.

Example 4.3. Below we list some $\{+, -\}$-labeled graphs with special names, which serve as motifs in the work of Aduddell et al. [2] and Tyson et al. [48] :

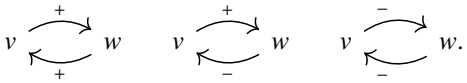
- **positve autoregulation** and **negative autoregulation**:



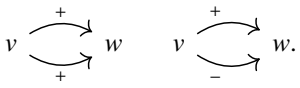
- **positive stimulation** and **negative stimulation**:



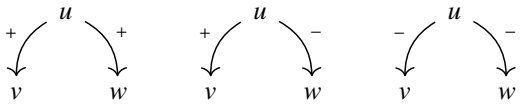
- the **positive feedback loop**, **negative feedback loop** and **double-negative feedback loop**:



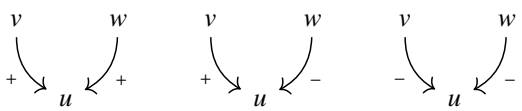
- the **coherent feedforward loop** and **incoherent feedforward loop**:



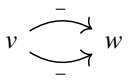
- three kinds of **branches**:



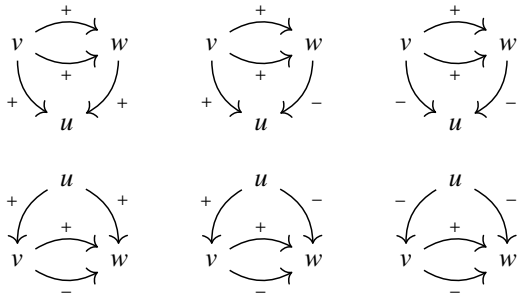
- three kinds of **logic gates**:



A third feedforward loop not mentioned by the above authors could be called the **double-negative feedforward loop**:



Starting from the fundamental motifs listed above, one can build other important ones, such as the **overlapping feedforward loops** formed by combining feedforward loops with branches or logic gates:



For a monoid M, denote the Kleisli category of M-labeled graphs (see Definition 4.2) by $\mathsf{K}(\mathsf{Gph}/GM)$. Given a homomorphism $\phi\colon M \to M'$ of monoids, there is a functor

$$\begin{array}{rccc} \phi_*\colon & \mathsf{K}(\mathsf{Gph}/G_M) & \to & \mathsf{K}(\mathsf{Gph}/G_{M'}) \\ \\ & (G,\ell) & \mapsto & (G, \phi\circ\ell) \\ & \Big(F\colon \mathsf{Free}_M(G,\ell_1) \to \mathsf{Free}_M(H,\ell_2)\Big) & \mapsto & \Big(F\colon \mathsf{Free}_{M'}(G,\phi\circ\ell_1) \to \mathsf{Free}_{M'}(H,\phi\circ\ell_2)\Big) \end{array}$$

which allows us to change labelings on the edges *functorially*, as we observe below:

Proposition 4.4. *The following assignment*

$$\begin{array}{rccc} F_{\mathrm{m}}\colon & \mathsf{Mon} & \to & \mathsf{Cat} \\ \\ & M & \mapsto & \mathsf{K}(\mathsf{Gph}/G_M) \\ & (M \xrightarrow{\phi} M') & \mapsto & (\mathsf{K}(\mathsf{Gph}/G_M) \xrightarrow{\phi_*} \mathsf{K}(\mathsf{Gph}/G_{M'}) \end{array}$$

defines a functor, where Mon *is the category of monoids.*

Proof. This follows because for any composable pair of monoid homomorphisms $M \xrightarrow{\phi} M' \xrightarrow{\phi'} M''$ we have

$$\phi'_*(G, \phi\circ\ell) = (G, \phi'\circ\phi\circ\ell).$$ □

Applying the Grothendieck construction to the functor in Proposition 4.4, we obtain the **category of monoid-labeled graphs and Kleisli morphisms** $\int F_{\mathrm{m}}$.

Proposition 4.5. $\int F_{\mathrm{m}}$ *is the category where:*

- *an object is a pair* $(M, p\colon G \to G_M)$*,*
- *a morphism from* $(M, p\colon G \to G_M)$ *to* $(M', p'\colon G' \to G_{M'})$ *is a pair* (ϕ, θ)*, where* $\phi\colon M \to M'$ *is a homomorphism of monoids and* $\theta\colon \mathsf{Free}(G) \to \mathsf{Free}(G')$ *is a functor such that this square commutes:*

$$\begin{array}{ccc} \mathsf{Free}(G) & \xrightarrow{\theta} & \mathsf{Free}(G') \\ \downarrow{\scriptstyle \tilde{p}} & & \downarrow{\scriptstyle \tilde{p}'} \\ \mathsf{B}M & \xrightarrow[\tilde{\phi}]{} & \mathsf{B}M' \end{array}$$

  where
  - *the functor* $\tilde{\phi}\colon \mathsf{B}M \to \mathsf{B}M' := \mathsf{B}(\phi)$*, where* $\mathsf{B}\colon \mathsf{Mon} \to \mathsf{Cat}$ *is the functor that takes a monoid to its associated 1-object category,*

– $\tilde{p}\colon \mathsf{Free}(G) \to \mathsf{B}M$ *is induced from from* $p\colon G \to G_M$*, which takes any morphism*

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{m-1}} v_{m-1} \xrightarrow{e_m} v_m$$

in $\mathsf{Free}(G)$ *to the unique morphism in* $\mathsf{B}M$ *associated to*

$$\ell(e_m)\cdots\ell(e_1) \in M,$$

– $\tilde{p}'\colon \mathsf{Free}(G') \to \mathsf{B}M'$ *is defined similarly to* $\tilde{p}$.

5. Commutative monoid-labeled graphs

Now we turn to graphs with edges labeled by elements of a *commutative* monoid. This allows for a new concept of morphism between labeled graphs, and permits a deeper study of feedback loops. We use additive notation for the operation in a commutative monoid.

Given a commutative monoid C, we define C-labeled graphs and maps of C-labeled graphs just as we do as when C is a mere set (Definitions 2.1 and 2.2). However, there is another useful concept of morphism between **finite** C-labeled graphs, meaning those with finitely many edges and vertices. (In applications, C-labeled graphs are usually finite.)

Definition 5.1. Let C be a commutative monoid, and let $(G, \ell) = (E, V, s, t, \ell)$ and $(G', \ell') = (E', V', s', t', \ell')$ be two finite C-labeled graphs. We define an **additive morphism** from (G, ℓ) to (G', ℓ') to be a map of graphs $f\colon G \to G'$ such that the following condition holds for all $e' \in E'$:

$$\ell'(e') = \sum_{\{e \in E\colon f_1(e) = e'\}} \ell(e). \tag{1}$$

The collection of finite C-labeled graphs and their additive morphisms forms a category whose composition law is induced from the composition law in the category FinGph. We denote this category by $C\mathsf{FinGph}$.

Equation (1) says that each edge e' of G' is labeled by the sum of the labels of edges of G that map to e'. We restrict ourselves to commutative monoids to make sure that the sum is independent of the order of summation, and to finite graphs to make sure the sum is a finite one. Furthermore, observe that $\ell(e') = 0$ if there is no $e \in E$ such that $f(e) = e'$, since any sum over the empty set vanishes.

We can push forward an C-labeling along any morphism of finite graphs. More precisely, we have the following:

Lemma 5.2. *Let C be a commutative monoid and (G, ℓ) a finite C-labeled graph. Then, for any map of finite graphs $f\colon G \to G'$, there is a unique C-labeling $f_*\ell$ of G', called the* **pushforward of ℓ along f***, such that f is an additive morphism of C-labeled graphs from (G, ℓ) to $(G', f_*\ell)$. This pushforward is covariantly functorial in the following sense:*

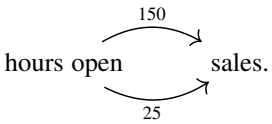
$$(f \circ g)_* = f_* \circ g_*, \qquad 1_* = 1.$$

Proof. By the definition of additive morphism we are forced to take

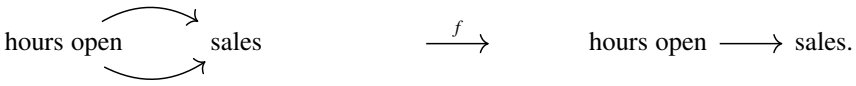
$$(f_*\ell)(e') := \sum_{\{e \in E\colon f_1(e) = e'\}} \ell(e)$$

for all edges e' of G', and this choice works. □

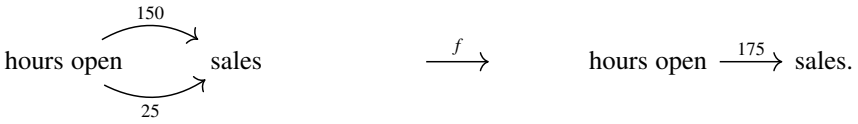
To see how this fact can be used in system dynamics or systems biology, suppose that each hour a coffee shop is open it sells \$150 of coffee and \$25 of tea. We can model this with an $(\mathbb{R}, +, 0)$-labeled graph:



We can simplify this model by mapping its underlying graph to a simpler graph:



Lemma 5.2 says there is a unique way to label the edges of the second graph that lets us lift the map f to an additive morphism of $(\mathbb{R}, +, 0)$-labeled graphs:



Just as Lemma 2.3 says that for any set L the forgetful functor from L-labeled graphs to graphs is a discrete fibration, Lemma 5.2 says that for any commutative monoid C the forgetful functor from C-labeled finite graphs to finite graphs is a discrete opfibration.

Proposition 5.3. *For any commutative monoid C, we have the following:*

(a) The forgetful functor $U\colon C\mathsf{FinGph} \to \mathsf{FinGph}$ is a discrete opfibration.

(b) The category of elements $\int Q$ of the covariant functor $Q\colon \mathsf{FinGph} \to \mathsf{Set}$ associated to U is equivalent to the category $C\mathsf{FinGph}$.

Proof. (a) was proved in Lemma 5.2. (b) follows from general principles, but one can see it concretely as follows. First note that the functor $Q\colon \mathsf{FinGph} \to \mathsf{Set}$ associated to the discrete fibration $U\colon C\mathsf{FinGph} \to \mathsf{FinGph}$ maps any finite graph to its set of C-labelings, and any map of finite graphs $f\colon G \to G'$ to the map sending C-labelings of G to C-labelings of G' given by Equation (1). Thus, the category of elements of Q is equivalent to $C\mathsf{FinGph}$. □

Proposition 5.3 (a) implies that there is a functorial way to push forward C-labelings along maps of graphs. We can also push forward labelings along a homomorphism of commutative monoids. Given a homomorphism $\phi\colon C \to C'$ of commutative monoid and an C-labeled finite graph (G, ℓ), we can define an C'-labeled graph

$$\phi_*(G, \ell) = (G, \phi \circ \ell). \tag{2}$$

Proposition 5.4. *The following assignment*

$$\begin{array}{rccl} F_{\mathrm{cm}}\colon & \mathsf{CommMon} & \to & \mathsf{Cat} \\ \\ & C & \mapsto & C\mathsf{FinGph} \\ & (C \xrightarrow{\phi} C') & \mapsto & (C\mathsf{FinGph} \xrightarrow{\phi_*} C'\mathsf{FinGph}) \end{array}$$

defines a functor, where CommMon *is the category of commutative monoids.*

The functors ϕ_* have many practical applications:

Example 5.5. Every commutative monoid C has a unique homomorphism $\phi\colon C \to 1$ where $\{1\}$ is the terminal monoid discussed in Example 3.1. The resulting functor

$$\phi_*\colon C\mathsf{FinGph} \to \{1\}\mathsf{FinGph} \cong \mathsf{Gph}$$

takes any C-labeled finite graph and discards the labeling, giving a finite graph. This can be used to discard information about *how* one vertex directly affects another and merely retain the fact *that* it does.

Example 5.6. There is a homomorphism $\phi\colon \mathbb{R} - \{0\} \to \{+, -\}$ from the multiplicative group of the reals (see Example 3.6) to the group $\{+, -\}$ (see Example 3.2) sending all positive numbers to $+$ and all negative numbers to $-$. The resulting functor ϕ_* turns *quantitative* information about how much one vertex directly affects another into purely *qualitative* information.

Example 5.7. There is a homomorphism $\phi\colon \mathbb{R} \to \{+, 0, -\}$ from the multiplicative monoid of the reals (see Example 3.7) to the monoid $\{+, 0, -\}$ (see Example 3.3) sending all positive numbers to $+$, all negative numbers to $-$, and 0 to 0. The resulting functor ϕ_* again turns quantitative information into qualitative information.

Example 5.8. The homomorphisms in Examples 5.5–5.7 all have right inverses. For example, there is a homomorphism $\psi\colon \{+, 0, -\} \to \mathbb{R}$ sending $+$ to 1, $-$ to -1 and 0 to 0, and this has

$$\phi \circ \psi = 1.$$

The functor ψ_* can be used to convert qualititative information about how one vertex directly affects another into quantitative information in a simple, default manner. Of course this should be taken with a grain of

salt: since $\psi \circ \phi \neq 1$, quantitative information that has been converted into qualitative information cannot be restored.

Applying the Grothendieck construction to the functor in Proposition 5.4 we obtain the **category of commutative monoid-labeled graphs** $\int F_{\mathrm{cm}}$.

Proposition 5.9. *$\int F_{\mathrm{cm}}$ is equivalent to the category where:*

- *an object is a triple (C, G, ℓ) where C is a commutative monoid and (G, ℓ) is an C-labeled graph;*
- *a morphism from (C, G, ℓ) to (C', G', ℓ') is a pair (ϕ, f), where $\phi \colon C \to C'$ is a homomorphism of commutative monoids and $f \colon G \to G'$ is a map of finite graphs such $f_*(\phi_* \ell) = \ell'$, where f_* is defined in Lemma 5.2 and ϕ_* is defined in Equation (2).*

6. Rig-labeled graphs

We have seen that passing from set-labeled graphs (Section 2) to monoid-labeled graphs (Section 3) lets us study of how one entity affects another indirectly through a path of edges, and gives a general way to analyze motifs (Section 4). When the labeling monoid is commutative, we can also define 'additive morphisms' between finite labeled graphs, which can be used to describe ways of simplifying labeled graphs (Section 5). In the commutative case we can also study feedback loops using homology theory (Section 8).

Given all this, it is mathematically tempting to study graphs whose edges are labeled by elements of a rig. A **rig** is a set R with the structure of both a commutative monoid $(R, +, 0)$ and a monoid $(R, \cdot, 1)$, obeying

$$r \cdot (s + t) = r \cdot s + r \cdot t, \qquad (r + s) \cdot t = r \cdot t + s \cdot t$$

$$0 \cdot r = 0 = r \cdot 0$$

for all $r, s, t \in R$. It has this name because it is a 'ring without negatives', or more precisely a ring that may not have negatives. The classic example is $\mathbb{N}$ with its usual addition and multiplication.

Despite the mathematically natural quality of rigs, their use of rig elements as polarities seems new and is somewhat speculative. What is the point of having two operations on the set of labels?

Example 6.1. The initial object in the category of rigs is $\mathbb{N}$ with its usual addition and multiplication. Given an $\mathbb{N}$-labeled graph, we can interpret an edge labeled by $n \in \mathbb{N}$

$$v \xrightarrow{n} w$$

as saying there are n ways for v to directly affect w. Given a path in an $\mathbb{N}$-labeled graph, say

$$v_0 \xrightarrow{n_1} v_1 \xrightarrow{n_2} v_2 \xrightarrow{n_3} \cdots \xrightarrow{n_k} v_k$$

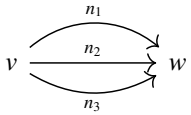
we can argue that there are $n_1 \cdots n_k$ ways for v_0 to affect v_k. This uses multiplication in the rig $\mathbb{N}$. Note also that using multiplication as the monoid operation we have a Kleisli morphism (Definition 4.2) from

$$v_0 \xrightarrow{n_1 \cdots n_k} v_k$$

to

$$v_0 \xrightarrow{n_1} v_1 \xrightarrow{n_2} v_2 \xrightarrow{n_3} \cdots \xrightarrow{n_k} v_k.$$

On the other hand, we can use addition to define additive morphisms (Definition 5.1). Then there is an additive morphism from



to

$$v \xrightarrow{n_1 + n_2 + n_3} w.$$

This is consistent with the idea that there are $n_1 + n_2 + n_3$ ways for v to affect w in the first $\mathbb{N}$-labeled graph, and that the second $\mathbb{N}$-labeled graph presents this information in a simplified manner.

Example 6.2. The boolean rig $\mathbb{B} = \{F, T\}$ has 'or' as addition, 'and' as multiplication, F as 0, and T as 1. Equivalently, we can take $\mathbb{B} = \{0, 1\}$ with the following addition and multiplication:

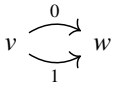
$+$	0	1
0	0	1
1	1	1

$\cdot$	0	1
0	0	0
1	0	1

We can use:

- the absence of an edge from v to w to mean that there is *no knowledge of whether* v directly affects w,
- an edge labeled by 0 from v to w to indicate the *known absence of a direct effect* of some specific sort of v on w,
- an edge labeled by 1 from v to w to indicate that there is *known presence of a direct effect* of some specific sort of v on w.

For example,



means it is known that v does not affect w in one way, but that it does affect it in some other way. We can simplify this using an additive morphism to

$$v \xrightarrow{1} w.$$

Example 6.3. We can generalize Example 6.2 as follows. A **quantale** is defined to be a poset Q with least upper bounds of arbitrary subsets, equipped with a multiplication $\cdot\colon Q \times Q \to Q$ that preserves least upper bounds in each argument. A quantale is **unital** if it has a unit 1 for the multiplication. Any unital quantale becomes a rig where the addition defined to be the binary join $\vee\colon Q \times Q \to Q$.

The boolean rig $\mathbb{B}$ is an example of a unital quantale, but there are many others. For example, $\mathbb{B}^n$ is a unitary quantale that allows us to generalize Example 6.2 to a situation where there are n researchers, each with their own research on whether one vertex affects another. Master has carried out a detailed study of networks using quantale theory [35, 36], which overlaps in interesting ways with our work here. Instead of considering an arbitrary graph with edges labeled by elements of a rig, she considers a set of vertices V together with a map $\ell\colon V \times V \to Q$ for some quantale Q. We can can think of this as a Q-labeled complete directed graph. In this approach the absence of an effect of $v \in V$ on $w \in V$ is indicated, not by the absence of an edge from v to w, but by setting $\ell(v, w) = 0$, where $0 \in Q$ is the bottom element (the least upper bound of the empty set).

We have seen that the multiplicative monoid of $\mathbb{Z}/3$ is useful for describing positive and negative effects as well as the absence of an effect (Example 3.3). While $\mathbb{Z}/3$ becomes a ring with its usual addition, this is problematic in the applications we are considering because $-1 + 1 = 0$, suggesting that a positive effect necessarily cancels a negative effect. To get around this we can introduce a new element, i, for 'indeterminate', and set $-1 + 1 = i$.

Example 6.4. There is a 4-element commutative rig $S = \{1, 0, -1, i\}$ with addition and multiplication given as follows:

$+$	1	0	-1	i
1	1	1	i	i
0	1	0	-1	i
-1	i	-1	-1	i
i	i	i	i	i

$\cdot$	1	0	-1	i
1	1	0	-1	i
0	0	0	0	0
-1	-1	0	1	i
i	i	0	i	i

We can use 1 to indicate a positive effect, -1 to indicate a negative effect, 0 to indicate the absence of an effect, and i to indicate an indeterminate effect: one that could be positive, negative or absent. The addition and multiplication rules in this rig capture the following intuitions:

- the sum of positive effects is positive, while the sum of negative effects is negative,
- the sum of a positive and a negative effect is indeterminate,
- the product of positive effects is positive, as is the product of negative effects,
- the product of a positive and a negative effect is negative,

- any operation applied to an indeterminate effect produces an indeterminate effect, except for multiplication by 0.

To check that S really is a rig, we can use a construction of Golan [21, Ex. 1.10]. For any monoid M, its power set PM becomes a rig with union as addition and with multiplication defined by

$$X \cdot Y = \{xy \mid x \in X, y \in Y\}$$

for $X, Y \in PM$. The rig S can then be seen as $P(\mathbb{Z}/2)$ using the following identifications, where we write the group $\mathbb{Z}/2$ multiplicatively as $\{+, -\}$ as in Example 3.2:

$$0 = \{\}, \qquad 1 = \{+\}, \qquad -1 = \{-\}, \qquad i = \{+, -\}.$$

Example 6.5. We can generalize Example 6.4 as follows. For any monoid M and any commutative unital quantale Q, the set Q^M becomes a rig where addition is defined pointwise:

$$(f + g)(x) = f(x) \vee g(x), \qquad x \in M,$$

where $\vee$ stands for the greatest lower bound, and multiplication is defined by a kind of convolution:

$$(f \cdot g)(x) = \bigvee_{\{y,z \in M \,|\, x=yz\}} f(y) \cdot g(z), \qquad x \in M.$$

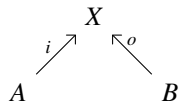
It is interesting to apply this construction to any quantale Q, thought of as describing 'generalized truth values', and any monoid M from Examples 3.1–3.11. Example 6.4 arises from taking $Q = \mathbb{B}$ and $M = \mathbb{Z}/2$.

Finally, there is the obvious example:

Example 6.6. The ring $\mathbb{R}$ with its usual addition and multiplication can be used to describe effects in a quantitative rather than purely qualitative way.

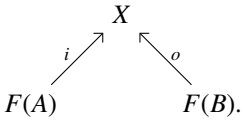
7. Open labeled graphs

Experience has shown that 'open' systems—systems that can interact with their environments—are well modeled using cospans [17]. A **cospan** in some category A is a diagram of this form:



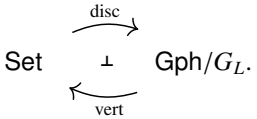
We call X the **apex**, A and B the **feet**, and i and o the **legs** of the cospan. The apex describes the system itself. The feet describe 'interfaces' through which the system can interact with the outside world. The legs describe how the interfaces are included in the system. If the category A has finite colimits, we can compose cospans using pushouts and tensor them using coproducts. Composition describes the operation of attaching two open systems together in series by identifying one interface of the first with one of the second. Tensoring describes setting open systems side by side, in parallel. Via these operations we obtain a symmetric monoidal double category with cospans in A as its horizontal 1-cells. Symmetric monoidal double categories take a while to get used to, but they nicely capture many of the operations available here.

However, we often want the systems to have more structure than their interfaces. We can sometimes model this using the theory of structured cospans. Here we start with a functor $F \colon \mathsf{A} \to \mathsf{X}$, typically the left adjoint of some functor $R \colon \mathsf{X} \to \mathsf{A}$ that forgets extra structure possessed by objects of X but not by those of A. Then a ***F*-structured cospan** is a diagram in X of the form:



When A and B have finite colimits and F preserves them, there is a symmetric monoidal double category where the horizontal 1-morphisms are F-structured cospans [6, 7, 14, 41].

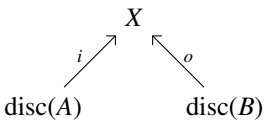
One of the simplest examples of this theory involves open L-labeled graphs for some set L. For this we use the adjoint functors



where

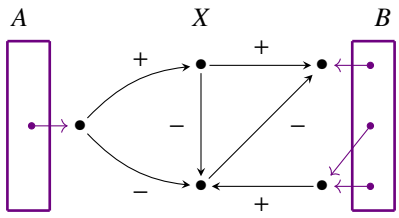
- $\mathrm{disc}\colon \mathsf{Set} \to \mathsf{Gph}/G_L$ takes a set S to the unique L-labeled graph with vertex set S and no edges;
- $\mathrm{vert}\colon \mathsf{Gph}/G_L \to \mathsf{Set}$ takes an L-labeled graph to its set of vertices.

Definition 7.1. Given a set L, an **open L-labeled graph** is a diagram in Gph/G_L of the form



for some sets A and B. We call this an open L-labeled graph **from** A **to** B.

For example, here is an open L-labeled graph with $L = \{+, -\}$:



A and B are sets, which we can think of as graphs with no edges using the functor disc. The purple arrows show how $\mathrm{disc}(A)$ and $\mathrm{disc}(B)$ are mapped to the L-graph X drawn in black.

The main point of open L-labeled graphs is that we can compose them to form bigger ones. To do this, we take an open L-labeled graph from A to B, and one from B to C, and create one from A to C by taking a pushout. This is formalized in the following already known result.

Theorem 7.2. *For any set L, there is a symmetric monoidal double category of open L-labeled graphs, $\mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_L)$, in which*

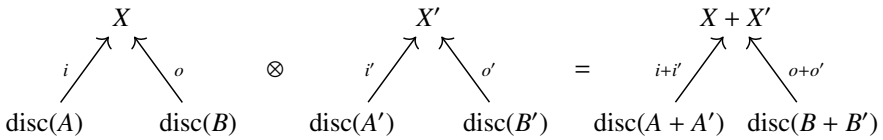
- *an object is a set,*
- *a vertical 1-morphism from A to B is a function $f\colon A \to B$,*
- *a horizontal 1-cell from A to B is an open L-labeled graph from A to B:*

$$\mathrm{disc}(A) \xrightarrow{\ i\ } X \xleftarrow{\ o\ } \mathrm{disc}(B),$$

- *a 2-morphism is a* **map of open L-labeled graphs**, *that is, a commutative diagram in Gph/G_L of the form*

$$\begin{array}{ccccc} \mathrm{disc}(A) & \xrightarrow{\ i\ } & X & \xleftarrow{\ o\ } & \mathrm{disc}(B) \\ {\scriptstyle \mathrm{disc}(f)}\downarrow & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \mathrm{disc}(g)} \\ \mathrm{disc}(A') & \xrightarrow[\ i'\ ]{} & X' & \xleftarrow[\ o'\ ]{} & \mathrm{disc}(B'). \end{array}$$

Vertical composition is done using composition in Set, *while horizontal composition is done using pushouts in* Gph/G_L. *The tensor product of two open L-labeled graphs is*



where $i + i'$ and $o + o'$ are defined using the fact that disc *preserves binary coproducts, and the tensor product of two 2-morphisms is given by:*

$$\begin{array}{ccccc} \mathrm{disc}(A_1) & \xrightarrow{i_1} & X_1 & \xleftarrow{o_1} & \mathrm{disc}(B_1) \\ {\scriptstyle \mathrm{disc}(f)}\downarrow & & {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \mathrm{disc}(g)} \\ \mathrm{disc}(A_2) & \xrightarrow[i_2]{} & X_2 & \xleftarrow[o_2]{} & \mathrm{disc}(B_2) \end{array} \quad \otimes \quad \begin{array}{ccccc} \mathrm{disc}(A_1') & \xrightarrow{i_1'} & X_1' & \xleftarrow{o_1'} & \mathrm{disc}(B_1') \\ {\scriptstyle \mathrm{disc}(f')}\downarrow & & {\scriptstyle \alpha'}\downarrow & & \downarrow{\scriptstyle \mathrm{disc}(g')} \\ \mathrm{disc}(A_2') & \xrightarrow[i_2']{} & X_2' & \xleftarrow[o_2']{} & \mathrm{disc}(B_2') \end{array}$$

$$= \begin{array}{ccccc} \mathrm{disc}(A_1 + A_1') & \xrightarrow{i_1 + i_1'} & X_1 + X_1' & \xleftarrow{o_1 + o_1'} & \mathrm{disc}(B_1 + B_1') \\ {\scriptstyle \mathrm{disc}(f + f')}\downarrow & & {\scriptstyle \alpha + \alpha'}\downarrow & & \downarrow{\scriptstyle \mathrm{disc}(g + g')} \\ \mathrm{disc}(A_2 + A_2') & \xrightarrow[i_2 + i_2']{} & X_2 + X_2' & \xleftarrow[o_2 + o_2']{} & \mathrm{disc}(B_2 + B_2'). \end{array}$$

Proof. The categories Set and Gph/G_L have finite colimits (see Proposition 2.4(a)) and $\mathrm{disc}\colon \mathsf{Set} \to \mathsf{Gph}/G_L$ preserves them, since it is a left adjoint. The theorem thus follows from the theory of structured cospans: see [6, Sec. 6.1] and [7, Sec. 6.2] for two different proofs. Using the same assumptions, Theorem 2.3 of [41] yields a stronger result: $\mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_L)$ is a cocartesian equipment, and thus a cocartesian double category. □

If M is a monoid we can define open M-labeled graphs as in Definition 7.1, not using the monoid structure, only the underlying set of M. However, we have the extra ability to convert any open M-labeled graph into an 'open M-graded category', which we now define. First, note that the categories of sets and M-labeled categories are related by a pair of adjoint functors

$$\mathsf{Set} \underset{\text{Vert}}{\overset{\text{Disc}}{\rightleftarrows}} \mathsf{Cat}/\mathsf{B}M.$$

where

- $\mathsf{Disc}\colon \mathsf{Set} \to \mathsf{Cat}/\mathsf{B}M$ takes a set S to the unique M-graded category with object set S and only identity morphisms;
- $\mathrm{Vert}\colon \mathsf{Cat}/\mathsf{B}M \to \mathsf{Set}$ takes an M-graded category to its set of objects.

We can apply the theory of structured cospans to the left adjoint Disc, as follows.

Definition 7.3. Given a monoid M, an **open M-graded category** is a diagram in $\mathsf{Cat}/\mathsf{B}M$ of the form

$$\begin{array}{ccc} & X & \\ {\scriptstyle i}\nearrow & & \nwarrow{\scriptstyle o} \\ \mathsf{Disc}(A) & & \mathsf{Disc}(B) \end{array}$$

for some sets A and B. We call this an open M-graded category **from** A **to** B.

Theorem 7.4. *For any monoid M, there is a symmetric monoidal double category of open M-labeled categories, $\mathbb{O}\mathbf{pen}(\mathsf{Cat}/\mathsf{B}M)$, in which*

- *an object is a set,*
- *a vertical 1-morphism from A to B is a function $f\colon A \to B$,*
- *a horizontal 1-cell from A to B is an open M-graded category from A to B:*

$$\mathsf{Disc}(A) \xrightarrow{i} X \xleftarrow{o} \mathsf{Disc}(B),$$

- *a 2-morphism is a* **map of open** M**-graded categories**, *that is, a commutative diagram in* $\mathsf{Cat}/\mathsf{B}M$ *of the form*

$$\begin{array}{ccccc} \mathsf{Disc}(A) & \xrightarrow{i} & X & \xleftarrow{o} & \mathsf{Disc}(B) \\ \downarrow{\scriptstyle \mathsf{Disc}(f)} & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \mathsf{Disc}(g)} \\ \mathsf{Disc}(A') & \xrightarrow[i']{} & X' & \xleftarrow[o']{} & \mathsf{Disc}(B'). \end{array}$$

Vertical composition is done using composition in Set, *while horizontal composition is done using pushouts in* $\mathsf{Cat}/\mathsf{B}M$. *The tensor product of two horizontal 1-cells is*

$$\begin{array}{ccccc} & X & & X' & \\ & {\scriptstyle i}\nwarrow\quad\nearrow{\scriptstyle o} & & {\scriptstyle i'}\nwarrow\quad\nearrow{\scriptstyle o'} & \\ \mathsf{Disc}(A) & & \mathsf{Disc}(B) & \mathsf{Disc}(A') & \mathsf{Disc}(B') \end{array} \quad \otimes \quad = \quad \begin{array}{ccc} & X + X' & \\ & {\scriptstyle i+i'}\nwarrow\quad\nearrow{\scriptstyle o+o'} & \\ \mathsf{Disc}(A + A') & & \mathsf{Disc}(B + B') \end{array}$$

where $i + i'$ *and* $o + o'$ *are defined using the fact that* Disc *preserves binary coproducts, and the tensor product of two 2-morphisms is given by:*

$$\begin{array}{ccccc} \mathsf{Disc}(A_1) & \xrightarrow{i_1} & X_1 & \xleftarrow{o_1} & \mathsf{Disc}(B_1) \\ \downarrow{\scriptstyle \mathsf{Disc}(f)} & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \mathsf{Disc}(g)} \\ \mathsf{Disc}(A_2) & \xrightarrow[i_2]{} & X_2 & \xleftarrow[o_2]{} & \mathsf{Disc}(B_2) \end{array} \quad \otimes \quad \begin{array}{ccccc} \mathsf{Disc}(A_1') & \xrightarrow{i_1'} & X_1' & \xleftarrow{o_1'} & \mathsf{Disc}(B_1') \\ \downarrow{\scriptstyle \mathsf{Disc}(f')} & & \downarrow{\scriptstyle \alpha'} & & \downarrow{\scriptstyle \mathsf{Disc}(g')} \\ \mathsf{Disc}(A_2') & \xrightarrow[i_2']{} & X_2' & \xleftarrow[o_2']{} & \mathsf{Disc}(B_2') \end{array}$$

$$= \quad \begin{array}{ccccc} \mathsf{Disc}(A_1 + A_1') & \xrightarrow{i_1 + i_1'} & X_1 + X_1' & \xleftarrow{o_1 + o_1'} & \mathsf{Disc}(B_1 + B_1') \\ \downarrow{\scriptstyle \mathsf{Disc}(f + f')} & & \downarrow{\scriptstyle \alpha + \alpha'} & & \downarrow{\scriptstyle \mathsf{Disc}(g + g')} \\ \mathsf{Disc}(A_2 + A_2') & \xrightarrow[i_2 + i_2']{} & X_2 + X_2' & \xleftarrow[o_2 + o_2']{} & \mathsf{Disc}(B_2 + B_2'). \end{array}$$

Proof. The categories Set and $\mathsf{Gph}/\mathsf{B}M$ have finite colimits (see Proposition 3.13(a)) and $\mathsf{Disc}\colon \mathsf{Set} \to \mathsf{Gph}/\mathsf{B}M$ preserves them, since it is a left adjoint. The theory of structured cospans thus implies that $\mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_L)$ is a symmetric monoidal double category as above. In fact [41, Thm. 2.3] implies that $\mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_L)$ is a cocartesian equipment. □

Now we construct a map that turns open M-labeled graphs into open M-graded categories. To do this we use Lemma 4.1, which says that there is an adjunction

$$\mathsf{Gph}/G_M \underset{\mathrm{Und}_M}{\overset{\mathsf{Free}_M}{\rightleftarrows}} \mathsf{Cat}/\mathsf{B}M. \qquad \bot$$

Theorem 7.5. *For any monoid* M *there is a symmetric monoidal double functor*

$$F_M\colon \mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_M) \to \mathbb{O}\mathbf{pen}(\mathsf{Cat}/\mathsf{B}M)$$

acting as follows:

- *on objects,* F_M *sends any set to itself,*
- *on vertical 1-morphisms,* F_M *sends any function to itself*
- *on horizontal 1-cells,* F_M *sends any open* M*-labeled graph*

$$\mathsf{Disc}(A) \xrightarrow{i} X \xleftarrow{o} \mathsf{Disc}(B),$$

to the open M*-graded category*

$$\mathsf{Disc}(A) \xrightarrow{i} \mathsf{Free}_M(X) \xleftarrow{o} \mathsf{Disc}(B),$$

- *on 2-morphisms, F_M sends any map of open M-labeled graphs*

$$\begin{array}{ccccc} \mathsf{Disc}(A) & \xrightarrow{i} & X & \xleftarrow{o} & \mathsf{Disc}(B) \\ \downarrow{\scriptstyle \mathsf{Disc}(f)} & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \mathsf{Disc}(g)} \\ \mathsf{Disc}(A') & \xrightarrow[i']{} & X' & \xleftarrow[o']{} & \mathsf{Disc}(B'). \end{array}$$

  to the map of open M-graded categories

$$\begin{array}{ccccc} \mathsf{Disc}(A) & \xrightarrow{i} & \mathsf{Free}_M(X) & \xleftarrow{o} & \mathsf{Disc}(B) \\ \downarrow{\scriptstyle \mathsf{Disc}(f)} & & \downarrow{\scriptstyle \mathsf{Free}_M(\alpha)} & & \downarrow{\scriptstyle \mathsf{Disc}(g)} \\ \mathsf{Disc}(A') & \xrightarrow[i']{} & \mathsf{Free}_M(X') & \xleftarrow[o']{} & \mathsf{Disc}(B'). \end{array}$$

Proof. Theorems 4.2 and 4.3 of [6] give a general way to construct maps between structured cospan double categories. The present theorem is a straightforward application of these, which only requires checking that the following diagram commutes up to natural isomorphism

$$\begin{array}{ccc} & & \mathsf{Gph}/G_M \\ & \overset{\mathrm{disc}}{\nearrow} & \downarrow{\scriptstyle \mathsf{Free}_M} \\ \mathsf{Set} & & \\ & \underset{\mathrm{disc}}{\searrow} & \\ & & \mathbf{Cat}/\mathsf{B}M \end{array}$$

and that all the arrows in this diagram preserve finite colimits (since they are left adjoints). □

Recall from Definition 4.2 that for any monoid M there is a category $\mathsf{K}(\mathsf{Gph}/G_M)$ of M-labeled graphs and Kleisli morphisms between these, where a Kleisli morphism from (G,ℓ) to (G',ℓ') is defined to be a map between free M-graded categories

$$f\colon \mathsf{Free}_M(G,\ell) \to \mathsf{Free}_M(G',\ell').$$

We can use the theory of structured cospans to define a double category of *open* M-labeled graphs and Kleisli morphisms. To do this, we use the functor

$$\Phi\colon \mathsf{Set} \to K(\mathsf{Gph}/G_M)$$

defined as follows:

- Φ sends any set S to the unique M-graded category with object set S and only identity morphisms. Note that this M-graded category is $\mathsf{Free}_M(G,\ell)$ where G is the graph with vertex set S and no edges, and ℓ is the only possible ℓ-labeling of G.
- Φ sends any function $f\colon S \to T$ to the unique map of M-graded categories acting as f on objects.

Theorem 7.6. *For any monoid M, there is a symmetric monoidal double category of open M-labeled graphs and Kleisli morphisms, $\mathbb{O}\mathbf{pen}(K(\mathsf{Gph}/G_M))$, in which*

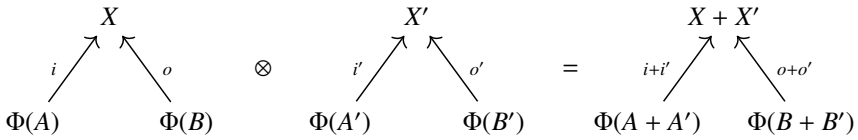
- *an object is a set,*
- *a vertical 1-morphism from A to B is a function $f\colon A \to B$,*
- *a horizontal 1-cell from A to B is an open M-labeled graph from A to B:*

$$\Phi(A) \xrightarrow{i} X \xleftarrow{o} \Phi(B),$$

- *a 2-morphism is a* **Kleisli morphism of open M-labeled graphs***, that is, a commutative diagram in $\mathsf{K}(\mathsf{Gph}/G_M)$ of the form*

$$\begin{array}{ccccc} \Phi(A) & \xrightarrow{i} & X & \xleftarrow{o} & \Phi(B) \\ \downarrow{\scriptstyle \Phi(f)} & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \Phi(g)} \\ \Phi(A') & \xrightarrow[i']{} & X' & \xleftarrow[o']{} & \Phi(B'). \end{array}$$

Vertical composition is done using composition in Set*, while horizontal composition is done using pushouts in* $\mathsf{K}(\mathsf{Gph}/G_M)$*. The tensor product of two open* M*-labeled graphs is*

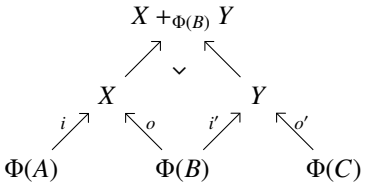


where $i + i'$ *and* $o + o'$ *are defined using the fact that* Φ *preserves binary coproducts, and the tensor product of two 2-morphisms is given by:*

$$\begin{array}{ccccc} \Phi(A_1) & \xrightarrow{i_1} & X_1 & \xleftarrow{o_1} & \Phi(B_1) \\ {\scriptstyle \Phi(f)}\downarrow & & {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \Phi(g)} \\ \Phi(A_2) & \xrightarrow[i_2]{} & X_2 & \xleftarrow[o_2]{} & \Phi(B_2) \end{array} \quad \otimes \quad \begin{array}{ccccc} \Phi(A_1') & \xrightarrow{i_1'} & X_1' & \xleftarrow{o_1'} & \Phi(B_1') \\ {\scriptstyle \Phi(f')}\downarrow & & {\scriptstyle \alpha'}\downarrow & & \downarrow{\scriptstyle \Phi(g')} \\ \Phi(A_2') & \xrightarrow[i_2']{} & X_2' & \xleftarrow[o_2']{} & \Phi(B_2') \end{array}$$

$$= \quad \begin{array}{ccccc} \Phi(A_1 + A_1') & \xrightarrow{i_1 + i_1'} & X_1 + X_1' & \xleftarrow{o_1 + o_1'} & \Phi(B_1 + B_1') \\ {\scriptstyle \Phi(f + f')}\downarrow & & {\scriptstyle \alpha + \alpha'}\downarrow & & \downarrow{\scriptstyle \Phi(g + g')} \\ \Phi(A_2 + A_2') & \xrightarrow[i_2 + i_2']{} & X_2 + X_2' & \xleftarrow[o_2 + o_2']{} & \Phi(B_2 + B_2'). \end{array}$$

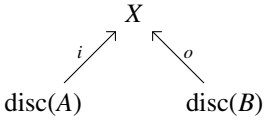
Proof. As mentioned in the proof of Theorem 7.2, a commonly stated theorem for constructing a symmetric monoidal double category of F-structured cospans assumes that we have categories A and X with finite colimits and a functor $F\colon \mathsf{A} \to \mathsf{X}$ that preserves them. We would like to apply this theorem to the functor $\Phi\colon \mathsf{Set} \to \mathsf{K}(\mathsf{Gph}/G_M)$. Unfortunately, while the category Set has finite colimits, Φ preserves them, and the Kleisli category $\mathsf{K}(\mathsf{Gph}/G_M)$ has finite coproducts, this Kleisli category does not have pushouts. Luckily, if one examines the proof of the commonly stated theorem, it becomes clear that we only need pushouts at one point: to compose structured cospans. In the case at hand, this means taking pushouts of this sort:



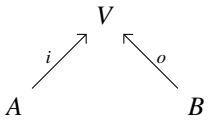
where X and Y are free M-graded categories. This pushout exists in $\mathsf{Cat}/\mathsf{B}M$, and the result is a free M-graded category, because in forming the pushout we do not need to identify any morphisms, only objects. It follows that this pushout also exists in $\mathsf{K}(\mathsf{Gph}/G_M)$. □

Finally, given a commutative monoid C, we can define a symmetric monoidal double category of open C-labeled finite graphs and additive morphisms. However, we cannot do this using structured cospans and the left adjoint $\mathsf{FinSet} \to \mathsf{CFinGph}$ to the functor assigning any C-labeled finite graph its set of vertices, since $\mathsf{CFinGph}$ does not have finite coproducts or pushouts, not even the limited class of pushouts that would be used to compose structured cospans. Thus, it seems we must invoke the theory of *decorated* cospans, introduced by Fong [16, 17] and later brought up to the level of double categories [7].

In this approach we view an open C-labeled finite graph



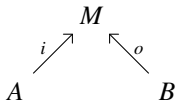
in a new way. We think of it as a cospan of finite sets



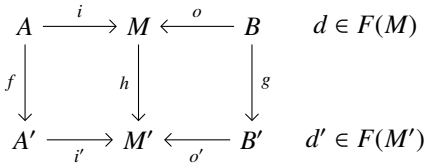
where V is 'decorated' with some extra stuff: namely, a C-labeled finite graph X having V as its set of vertices. We think of this 'decoration' X as an object of the category $F(V)$, where $F\colon \mathsf{FinSet} \to \mathbf{Cat}$ maps each finite set to the category of C-labeled finite graphs with that set of vertices, and additive morphisms between these.

In general, the theory of decorated cospans gives a symmetric monoidal double category $F\mathbb{C}\mathbf{sp}$ from a category A with finite colimits and a symmetric lax monoidal pseudofunctor $F\colon (\mathsf{A}, +) \to (\mathbf{Cat}, \times)$. In this double category $F\mathbb{C}\mathbf{sp}$:

- an object is an object of A;
- a vertical 1-morphism is a morphism of A;
- a horizontal 1-cell is an **F-decorated cospan**, that is, a diagram in A of the form



  together with a **decoration** $d \in F(M)$;
- a 2-morphism is a **map of F-decorated cospans**, that is, a commutative diagram in A of the form

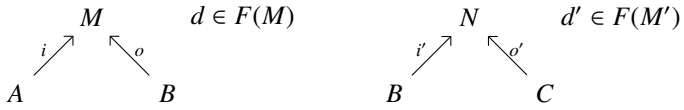


  together with a **decoration morphism** $\tau\colon F(h)(d) \to d'$ in $F(M')$.

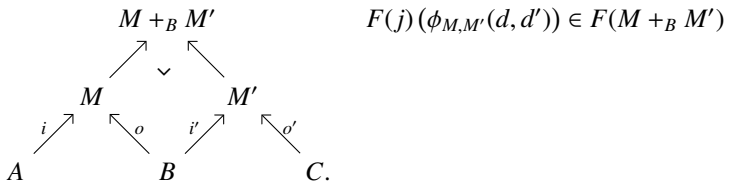
A key piece of the symmetric lax monoidal pseudofunctor F is its 'laxator', which gives for each pair of objects $M, M' \in \mathsf{A}$ a functor

$$\phi_{M,M'}\colon F(M) \times F(M') \to F(M + M').$$

This is used to compose F-decorated cospans, as follows. Given a composable pair of F-decorated cospans



we define their composite to be



Here the cospans are composed using a pushout in A, while the decoration is defined using the laxator and $j\colon M + M' \to M +_B M'$, the canonical map from the coproduct to the pushout. For details see [7, Sec. 2].

To define the double category of C-labeled finite graphs using this machinery we take $\mathsf{A} = \mathsf{FinSet}$. We want a decoration of $V \in \mathsf{FinSet}$ to be a C-labeled finite graph with V as its set of vertices. Thus we let $F\colon (\mathsf{FinSet}, +) \to (\mathbf{Cat}, \times)$ assign to V the category $F(V)$ where:

- An object is a C-labeled finite graph with vertex set V, say (V, E, s, t, ℓ).
- A morphism is an additive morphism of C-labeled graphs that is the identity on vertices.

We let F assign to each function $g\colon V \to V'$ the functor $F(g)\colon F(V) \to F(V')$ where:

- $F(g)$ sends any object (V, E, s, t, ℓ) to $(V', E, g \circ s, g \circ t, \ell)$.

- $F(g)$ sends any morphism $f\colon (V, E, s, t, \ell) \to (V, E', s', t', \ell')$ in $F(V)$, which is determined by a map of graphs $1_V\colon V \to V$, $f_1\colon E \to E'$, to the morphism $F(g)(f)\colon (V, E, f_1 \circ s, f_1 \circ t, \ell) \to (V', E', f_1 \circ s, f_1 \circ t, \ell')$ in $F(V')$ determined by the map of graphs $1_{V'}\colon V' \to V'$, $f_1\colon E \to E'$.

To parse this, it is useful to review the definition of 'additive morphism', Definition 5.1.

Next we must equip F with the structure of a lax monoidal pseudofunctor [7]. We omit some details here. The most important structure is the laxator

$$\phi_{V,V'}\colon F(V) \times F(V') \to F(V + V').$$

On objects, this takes a C-labeled finite graph with vertex set V, say (V, E, s, t, ℓ), and a C-labeled finite graph with vertex set V', say (V', E', s', t', ℓ'), and gives the C-labeled finite graph

$$(V + V', E + E', s + s', t + t', \langle \ell, \ell' \rangle)$$

with vertex set $V + V'$. Here $\langle \ell, \ell' \rangle\colon E + E' \to C$ is defined to equal ℓ on E and ℓ' on E'.

Given that $F\colon (\mathsf{FinSet}, +) \to (\mathbf{Cat}, \times)$ is a symmetric lax monoidal pseudofunctor, the theory of decorated cospans gives this result:

Theorem 7.7. *For any commutative monoid C, there is a symmetric monoidal double category of open C-labeled finite graphs and additive morphisms, $\mathbb{O}\mathbf{pen}(C\mathsf{FinGph})$, in which*

- *an object is a finite set;*
- *a vertical 1-morphism from A to B is a function $f\colon A \to B$;*
- *a horizontal 1-cell from A to B is an open C-labeled finite graph from A to B:*

$$A \xrightarrow{\ i\ } M \xleftarrow{\ o\ } B \qquad d \in F(M);$$

- *a 2-morphism is an* **additive morphism of open C-labeled graphs**, *that is, a commutative diagram in $C\mathsf{FinGph}$ of the form*

$$\begin{array}{ccccc} A & \xrightarrow{\ i\ } & M & \xleftarrow{\ o\ } & B \\ {\scriptstyle f}\downarrow & & {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle g} \\ A' & \xrightarrow[\ i'\ ]{} & M' & \xleftarrow[\ o'\ ]{} & B' \end{array} \qquad \begin{array}{l} d \in F(M) \\ \\ d' \in F(M'). \end{array}$$

Vertical composition is done using composition in FinSet*, while horizontal composition is done using composition of F-decorated cospans.*

Proof. The double category structure follows from [7, Thm. 2.1]. The symmetric monoidal structure, which we have not described here, follows from [7, Thm. 2.2], and is detailed there. □

8. Feedback loops and homology

In system dynamics, graphs labeled by signs are often used to study feedback loops. We can approach this systematically for graphs labeled by elements of a commutative monoid using homology theory. Homology lets us detect cycles in a graph. The homology of a graph with coefficients in an abelian group is well understood, and independent of the direction of the graph's edges, but in fact we can define the homology of a graph with coefficients in a commutative monoid, in a way that depends on the edge directions. We are most interested in taking coefficients in $\mathbb{N}$, because this gives a way to study feedback loops.

Let C be a commutative monoid. In what follows we use $C[X]$ to stand for the set of formal finite linear combinations of elements of X, with coefficients in C. Thus,

$$C[X] = \left\{ \sum_{x \in X} a_x x \;\middle|\; a_x \in C, \text{ all but finitely many } a_x \text{ are zero} \right\}.$$

This is a commutative monoid under addition. In particular, if we make $\mathbb{N}$ into a commutative monoid with addition as the monoid operation, $\mathbb{N}[X]$ is the free commutative monoid on X.

Definition 8.1. Given a graph $G = (E, V, s, t)$ and a commutative monoid C, define

$$C_0(G, C) = C[V], \qquad C_1(G, C) = C[E].$$

We call $C_0(G, C)$ (resp. $C_1(G, C)$) the commutative monoid of **0-chains** (resp. **1-chains**) on G with coefficients in C.

The source and target maps of the graph give two maps sending 1-chains to 0-chains. Namely, starting from the maps $s, t\colon E \to V$ we can extend them by linearity to monoid homomorphisms

$$C[s], C[t]\colon C_1(G, C) \to C_0(G, C).$$

Indeed, there is a functor $C[-]\colon \mathsf{Set} \to \mathsf{CommMon}$. We have already defined this on objects, and given a map of sets $f\colon X \to Y$ we define $C[f]\colon C[X] \to C[Y]$ by

$$C[f]\left(\sum_{x\in X} a_x x\right) = \sum_{x\in X} a_x f(x).$$

When C is an abelian group we can define a group homomorphism $d\colon C_1(G, C) \to C_0(G, C)$ by

$$d = C[s] - C[t].$$

In this context the usual expressions for homology groups as quotient groups simplify, and we can define

$$H_1(G, C) = \ker d, \qquad H_0(G, C) = \operatorname{coker} d.$$

But when working with commutative monoids that are not abelian groups, we cannot form the difference $C[s] - C[t]$, so we use the equalizer and coequalizer of $C[s]$ and $C[t]$ rather than the kernel and cokernel of their difference, defining

$$H_1(G, C) = \operatorname{eq}(C[s], C[t]), \qquad H_0(G, C) = \operatorname{coeq}(C[s], C[t]).$$

More formally:

Definition 8.2. Given a graph G and a commutative monoid C, we define $H_1(G, C)$, the **first homology** of G with coefficients in C, to be the equalizer of the morphisms

$$C_1(G, C) \underset{C[t]}{\overset{C[s]}{\rightrightarrows}} C_0(G, C).$$

We call an element of $H_1(G, C)$ a **1-cycle** or simply a **cycle** with coefficients in C. We define $H_0(G, C)$, the **zeroth homology** of G with coefficients in C, to be the coequalizer of s and t.

We can fit everything into one diagram as follows:

$$H_1(G, C) \xrightarrow{i} C_1(G, C) \underset{C[t]}{\overset{C[s]}{\rightrightarrows}} C_0(G, C) \xrightarrow{p} H_0(G, C).$$

where i is the inclusion of the equalizer of $C[s]$ and $C[t]$ and p is the projection onto their coequalizer. When C is an abelian group, we can rewrite this as an exact sequence

$$0 \to H_1(G, C) \xrightarrow{i} C_1(G, C) \xrightarrow{d} C_0(G, C) \xrightarrow{p} H_0(G, C) \to 0.$$

Example 8.3. This graph:

$$G = \quad u \quad v$$

has $H_1(G, \mathbb{N}) \cong \mathbb{N}$, while this graph

$$G' = \quad u \quad v$$

has $H_1(G', \mathbb{N}) \cong \{0\}$. In simple terms, this says that the first graph offers the opportunity for a feedback loop, while the second does not. On the other hand, we have $H_1(G, \mathbb{Z}) \cong H_1(G', \mathbb{Z}) \cong \mathbb{Z}$. We have $H_0(G, \mathbb{N}) \cong H_0(G, \mathbb{N}') \cong \{0\}$ and similarly with $\mathbb{Z}$ coefficients, since both graphs are connected.

The zeroth homology of a graph with coefficients in a commutative monoid is easily understood. There are various concepts of 'connected component' for a graph, but define the set of **undirected components** of a graph G, $\pi_0(G)$, to be the coequalizer

$$E \underset{t}{\overset{s}{\rightrightarrows}} V \xrightarrow{p} \pi_0(G). \tag{3}$$

Thus, two vertices v, v' lie in the same undirected component iff there is an **undirected path** from v to v': a sequence of vertices $v = v_0, \ldots, v_n = v'$ such that for each $i = 1, \ldots, n$ there is either an edge $e\colon v_{i-1} \to v_i$ or an edge $e\colon v_i \to v_{i-1}$.

Theorem 8.4. *Let G be a graph and C a commutative monoid. Then $H_0(G, C) \cong C[\pi_0(G)]$.*

Proof. This can be shown directly, but we sketch a proof using some facts, well known for abelian groups, which have easy analogues for commutative monoids [37] and even more general algebraic structures [28]. First, given $A, B \in \mathsf{CommMon}$, the set of homomorphisms $A \to B$ can be made into a commutative monoid $[A, B]$ using pointwise operations:

$$(f + g)(a) = f(a) + g(a), \qquad \forall f, g \in [A, B],\ a \in A.$$

Second, we can define a tensor product $A \otimes B$ of commutative monoids such that homomorphisms $A \otimes B \to C$ correspond naturally to maps $A \times B \to C$ that are homomorphisms in each argument. This tensor product obeys hom-tensor adjointness: that is, there is a natural isomorphism

$$\mathsf{CommMon}(A \otimes B, C) \cong \mathsf{CommMon}(B, [A, C]).$$

For any commutative monoid C there is a natural isomorphism

$$C[-] \cong C \otimes \mathbb{N}[-]$$

of functors from $\mathsf{CommMon}$ to Set. The functor $\mathbb{N}[-]$ is a left adjoint to the forgetful functor from $\mathsf{CommMon}$ to Set, and the functor $C \otimes -$ is left adjoint to the functor $[C, -]$. It follows that $C[-] \cong C \otimes \mathbb{N}[-] \colon \mathsf{Set} \to \mathsf{CommMon}$ is a left adjoint, so it preserves colimits. Applying this functor to the coequalizer diagram (3), we get a coequalizer diagram

$$C[E] \underset{C[t]}{\overset{C[s]}{\rightrightarrows}} C[V] \xrightarrow{C[t]} C[\pi_0(G)].$$

On the other hand, we have defined $H_0(G, C)$ as the coequalizer

$$C[E] \underset{C[t]}{\overset{C[s]}{\rightrightarrows}} C[V] \xrightarrow{C[t]} H_0(G, C)$$

though we have denoted $C[E]$ as $C_1(G, C)$ and $C[V]$ as $C_0(G, C)$. It follows that $H_0(G) \cong C[\pi_0(G)]$. □

Now let us turn to the first homology of a graph G with coefficients in a commutative monoid. When C is an abelian group, $H_1(G, C)$ is isomorphic to the homology of the graph G viewed as a topological space, which is well understood [34, Sec. III.3]. For example, $H_1(G, \mathbb{Z})$ is a free abelian group whose rank is the genus of the graph. More generally, whenever C is an abelian group, the universal coefficient theorem implies

$$H_1(G, C) \cong C \otimes_{\mathbb{Z}} H_1(G, \mathbb{Z}).$$

The novelty lies in the case when C is *not* a group, since then the directions of the edges matter, as in Example 8.3. We are then doing a simple sort of 'directed algebraic topology'. We are mainly interested in the case $C = \mathbb{N}$, since this captures the structure of possible feedback loops in the graph. We now turn toward analyzing the structure of $H_1(G, \mathbb{N})$. As we shall seein Example 8.7, it is not always a free commutative monoid.

To begin, note that there is a canonical preorder on any commutative monoid C, given by

$$x \le y \iff x + a = y \text{ for some } a \in C.$$

If C is an abelian group then $x \le y$ for all $x, y \in C$. But for $C = \mathbb{N}$, the canonical preorder is the usual linear ordering on natural numbers. More generally, in the free commutative monoid $\mathbb{N}[X]$ on any set X the canonical preorder is given by

$$\sum_{x \in X} a_x x \le \sum_{x \in X} B_x x \iff a_x \le B_x \text{ for all } x \in X.$$

Definition 8.5. Given a commutative monoid C, an element $x \in C$ is **minimal** if it is nonzero and $y \le x$ implies that $y = x$ or $y = 0$.

Any free commutative monoid $\mathbb{N}[X]$ is freely generated by its minimal elements, which correspond to the elements of X. A weaker statement holds for the first homology of a graph with coefficients in $\mathbb{N}$.

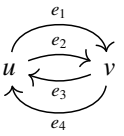
Theorem 8.6. *For any graph G, $H_1(G, \mathbb{N})$ is generated by its minimal elements.*

Proof. First recall that $H_1(G, \mathbb{N})$ is a submonoid of $C_1(G, \mathbb{N}) \cong \mathbb{N}[E]$ where E is the set of edges of G. The canonical preorder on $H_1(G, \mathbb{N})$ is a restriction of that in $C_1(G, \mathbb{N})$, since given $x, y \in H_1(G, \mathbb{N})$ there exists $z \in C_1(G, \mathbb{N})$ with $x + z = y$ if and only if there exists $z \in H_1(G, \mathbb{N})$ obeying this equation. As a result, since

there are no infinite descending chains of elements $c_1 > c_2 > c_3 > \cdots$ in $C_1(G, \mathbb{N}) \cong \mathbb{N}[E]$, there are no infinite descending chains in $H_1(G, \mathbb{N})$.

Next we use this to show that any $c \in H_1(G, \mathbb{N})$ is a sum of minimal elements. If c is minimal, we are done. Otherwise we can write c as a sum of two smaller but nonzero elements. If these are both minimal then we are done; if not, we can express either one that is not minimal as a sum of two smaller nonzero elements. If we could keep repeating the process of breaking each non-minimal element into a sum of two smaller nonzero elements forever, then $H_1(G, \mathbb{N})$ would have an infinite descending chain. But since this is impossible, the process must terminate, expressing c as a finite sum of minimal elements. □

Example 8.7. Consider the graph

$Q =$ 

One can see that the minimal elements of $H_1(Q, \mathbb{N})$ are the cycles

$$a = e_1 + e_3, \quad b = e_1 + e_4, \quad c = e_2 + e_3, \quad d = e_2 + e_4.$$

Thus these generate $H_1(Q, \mathbb{N})$, but not freely, since $a + c = b + d$. Moreover, note that a free commutative monoid is always freely generated by its minimal elements, since for any set S, the minimal elements of the free commutative monoid $\mathbb{N}[S]$ are precisely the linear combinations of elements of S where one coefficient equals 1 and all the rest are zero. Thus, $H_1(Q, \mathbb{N})$ is not a free commutative monoid.

We call the minimal elements of $H_1(G, \mathbb{N})$ **minimal cycles**. We are interested in them because they determine the feedback around all cycles in G. Let us make this precise. For any commutative monoid C and any C-labeled graph (G, ℓ), there is a map

$$C_1(G, \mathbb{N}) \to C$$

defined by

$$\sum_{e \in E} n_e e \mapsto \sum_{e \in E} n_e \ell(e).$$

Indeed this is the unique homomorphism of commutative monoids extending ℓ from E to the free commutative monoid on E, namely $C_1(G, \mathbb{N})$. If we restrict this map to cycles we get a homomorphism that we call

$$\tilde{\ell} \colon H_1(G, \mathbb{N}) \to C.$$

Given a cycle $c \in H_1(G, \mathbb{N})$, we call $\tilde{\ell}(c)$ the **feedback** of (G, ℓ) around c.

Corollary 8.8. *For any commutative monoid C and any C-labeled graph (G, ℓ), the homomorphism $\tilde{\ell} \colon H_1(G, \mathbb{N}) \to C$ is determined by its values on minimal cycles.*

Proof. This follows from Theorem 8.6: minimal cycles generate $H_1(G, \mathbb{N})$. □

While they are defined in an order-theoretic way, minimal cycles have an appealing 'geometrical' characterization as homology classes of certain 'simple' loops in G.

Definition 8.9. Let G be a graph. A **path** in G is a finite sequence of edges $e_1, \ldots e_n$ such that $t(e_i) = s(e_{i+1})$ for $i = 1, \ldots, n-1$. An **loop** is an path that ends where it starts, meaning $s(e_1) = t(e_n)$.

We can denote any path in G as follows:

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_n$$

and a loop as follows:

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_0$$

Note from Section 4 that the paths in a graph G are precisely the morphisms in the free category on G.

Any path γ in G gives an element of $[\gamma] \in C_1(G, \mathbb{N})$, defined to be the sum of that path's edges. That is, if

$$\gamma = \left(v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_n \right)$$

then we define

$$[\gamma] = e_1 + \cdots + e_n.$$

If γ is a loop then $[\gamma]$ is a cycle since then

$$s(e_1) + \cdots + s(e_n) = t(e_1) + \cdots + t(e_n).$$

Definition 8.10. Two loops γ, δ in a graph G are **homologous** if $[\gamma] = [\delta]$.

Definition 8.11. A loop

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_0$$

is **simple** if all the vertices $v_0, \ldots, v_{n-1}$ are distinct.

It is easy to see that if a loop γ is not simple, we can chop it where it crosses itself, obtaining two loops δ and η such that

$$[\gamma] = [\delta] + [\eta]$$

and $[\delta], [\eta] \neq 0$. Thus, $[\gamma]$ cannot be minimal if γ is not simple. In fact a much stronger result holds:

Theorem 8.12. *For any graph G, there is a bijection between homology classes of simple loops in G and minimal cycles in G, which sends all loops equivalent to the simple loop γ to the cycle $[\gamma]$.*

Proof. This follows from Lemmas 8.13 and 8.14 below. □

Lemma 8.13. *If γ is a simple loop in a graph G then $[\gamma]$ is a minimal cycle.*

Proof. Consider a simple loop

$$\gamma = \left(v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_0\right).$$

Then

$$[\gamma] = \sum_{i=1}^{n} e_i.$$

Any cycle $c \leq [\gamma]$ must be a chain less than or equal to c in $C_1(G, \mathbb{N})$, and since $C_1(G, \mathbb{N})$ is the free commutative monoid on the set of edges of G, any chain $c \leq [\gamma]$ must be of the form

$$c = \sum_{i \in S} e_i.$$

where $S \subseteq \{1, \ldots, n\}$. We have

$$\mathbb{N}[s](c) = \sum_{i \in S} v_{i-1}, \quad \mathbb{N}[t](c) = \sum_{i \in S} v_i.$$

For c to be a cycle we must have $\mathbb{N}[s](c) = \mathbb{N}[t](c)$. But since all the vertices $v_1, \ldots, v_n$ are distinct (while $v_0 = v_n$), the two sums above can only be equal if S is all of $\{1, \ldots, n\}$, in which case $c = [\gamma]$, or S is empty, in which case $c = 0$. (Note that since $C_0(G, \mathbb{N})$ is free on the set of vertices of G, the two sums can only be equal if they are 'visibly' equal: there are no extra relations.) Thus $[\gamma]$ is minimal. □

Lemma 8.14. *For each minimal cycle c in a graph G there exists a simple loop γ in G such that $[\gamma] = c$.*

Proof. Let c be a minimal cycle. It is nonzero, so choose an edge with $e_1 \leq c$ in $C_1(G, \mathbb{N})$. Denote this edge as $v_0 \xrightarrow{e_1} v_1$. If $v_1 = v_0$ the path $v_0 \xrightarrow{e_1} v_1$ is a loop, say γ, so $[\gamma] = e_1$ is a nonzero cycle, so by the minimality of c we must have $c = [\gamma]$ and we are done. If on the other hand $v_1 \neq v_0$ then e_1 is not a cycle, so c must be the sum of e_1 and one or more edges, and at least one of these edges must have source v_1, since otherwise it would be impossible to have $s(c) = t(c)$. Choose one such edge and call it $v_1 \xrightarrow{e_2} v_2$. We now have a path

$$\delta = \left(v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2\right)$$

with $\delta \leq c$ and with v_0, v_1 distinct.

We continue along these lines by carrying out an inductive procedure. Assume we have a path

$$\delta = \left(v_0 \xrightarrow{e_1} \cdots \xrightarrow{e_n} v_n\right)$$

with $[\delta] \leq c$ and $v_0, \ldots, v_{n-1}$ distinct. If v_n equals any of the previous vertices, say $v_n = v_i$ with $0 \leq i < n$, then we obtain a simple loop

$$\gamma = \left(v_i \xrightarrow{e_{i+1}} \cdots \xrightarrow{e_n} v_n\right).$$

Since $[\gamma]$ is a nonzero cycle less than or equal to $[\delta]$ and thus c, by the minimality of c we must have $c = [\gamma]$ and we are done. Otherwise all the vertices $v_0, \ldots, v_n$ are distinct, so $[\delta]$ is not a cycle, so c must be a sum

of $[\delta]$ and one or more edges, and at least one of these edges must have source v_n, since otherwise it would be impossible to have $s(c) = t(c)$. Choose one such edge and denote it by $v_n \xrightarrow{e_{n+1}} v_{n+1}$. We now have a path

$$\delta' = \left(v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_n \xrightarrow{e_{n+1}} v_{n+1}\right)$$

with $[\delta'] \leq c$ and $v_0, \ldots, v_n$ distinct.

Since c is a finite sum of edges this procedure must eventually terminate: i.e., eventually v_k must equal one of the previous vertices $v_0, \ldots, v_{k-1}$, which are themselves all distinct. We thus obtain a simple loop γ giving a nonzero cycle $[\gamma] \leq c$, and by the minimality of c we must have $[\gamma] = c$. □

Having established a bijection between minimal cycles and homology classes of simple loops, we naturally want to know when two simple loops are homologous. It turns out that the only way to obtain a loop homologous to a simple loop is to change where the loop starts. More precisely:

Proposition 8.15. *Every loop homologous to a simple loop*

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_0$$

is of the form

$$v_k \xrightarrow{e_{k+1}} v_{k+1} \xrightarrow{e_{k+2}} \cdots \xrightarrow{e_{n+k-1}} v_{n+k-1} \xrightarrow{e_{n+k}} v_k$$

where we treat the subscripts as elements of $\mathbb{Z}/n$ and do addition mod n.

Proof. Suppose

$$\gamma = \left(v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} v_{n-1} \xrightarrow{e_n} v_0\right)$$

is a simple loop and

$$\delta = \left(w_0 \xrightarrow{f_1} w_1 \xrightarrow{f_2} \cdots \xrightarrow{f_{m-1}} w_{m-1} \xrightarrow{f_m} w_0\right)$$

is a loop homologous to γ, so

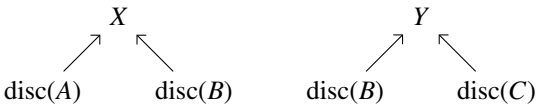
$$e_1 + \cdots + e_n = f_1 + \cdots + f_m.$$

Since γ is simple, all the vertices $v_0, \ldots, v_{n-1}$ are distinct, so the edges $e_1, \ldots, e_n$ are distinct. We must thus have $m = n$, with the list of edges $f_1, \ldots, f_n$ being some permutation of the list of edges $e_1, \ldots, e_n$. Since all the vertices $v_0, \ldots, v_{n-1}$ are distinct, the only permutations that make δ into a loop are cyclic permutations. □

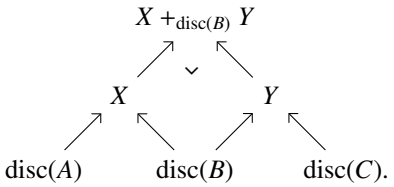
9. Emergent feedback loops

When we glue together two graphs, the resulting graph can have loops that are not contained in either of the original graphs. These are called 'emergent loops'. In applications, these represent new possible feedback loops that arise when we combine two systems. Since feedback loops are fundamental in system dynamics, it is important to pay close attention to this phenomenon.

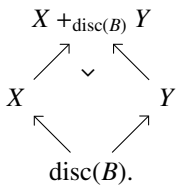
There are several ways to study emergent loops. The first is to study emergent paths in graphs formed by composing open graphs. We described a double category $\mathbb{O}\mathbf{pen}(\mathsf{Gph}/G_L)$ of open L-labeled graphs in Theorem 7.2, and we can define the double category of **open graphs**, $\mathbb{O}\mathbf{pen}(\mathsf{Gph})$, to be the special case where L is the one-element set, so the labeling becomes trivial. To compose two open graphs:



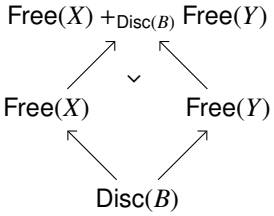
we take their pushout over a discrete graph on some set B:



To study paths in the pushout graph, let us simplify the diagram to the relevant part:



Paths in the pushout graph are morphisms in the free category $\mathsf{Free}(X +_{\mathrm{disc}(B)} Y)$. The left adjoint functor $\mathsf{Free}\colon \mathsf{Gph} \to \mathsf{Cat}$ preserves pushouts, so we can apply this functor to the above diagram and obtain the following pushout diagram:



since $\mathsf{Disc}(B) \cong \mathsf{Free}(\mathrm{disc}(B))$. For convenience let us make the abbreviation

$$\mathsf{C} = \mathsf{Free}(X) +_{\mathsf{Disc}(B)} \mathsf{Free}(Y).$$

Then the set of paths from a vertex v of the pushout graph to the vertex w is the homset $\mathsf{C}(v, w)$. We would like to understand which of these paths are 'emergent', i.e., not already paths in X or Y.

Let $M\{x, y\}$ be the free monoid on two generators x and y. Then the category C has a unique $M\{x, y\}$-grading where each edge of X has grade x and each edge of Y has grade y. Let $I\{x, y\}$ be the quotient of the monoid $M\{x, y\}$ by relations saying that x and y are idempotent:

$$x^2 = x, \quad y^2 = y.$$

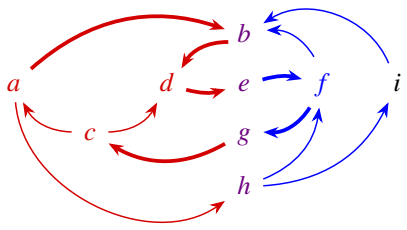
Bt Proposition 4.4, the quotient map $M\{x, y\} \to I\{x, y\}$ lets us push forward the $M\{x, y\}$-grading on C to obtain an $I\{x, y\}$-grading on this category. This has a simplifying effect: now the grade of a path records only how it goes back and forth between the graphs X and Y.

We can write

$$\mathsf{C}(v, w) = \bigsqcup_{m \in I\{g,h\}} \mathsf{C}_m(v, w)$$

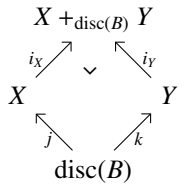
where $\mathsf{C}_m(v, w)$ is the set of paths of grade m from v to w. If v and w are vertices in X, the paths in $\mathsf{C}_x(v, w)$ are those that already existed in X, while all the other paths in $X(v, w)$ are 'emergent'. Similarly, if v and w are vertices in Y, all the paths not in $\mathsf{C}_y(v, w)$ are emergent. The grading gives a measure of the 'amount of emergence' involved in each path.

Example 9.1. Suppose X is the graph in red below while Y is the graph in blue, and B is the set of purple vertices:



If we treat the category C as $M\{x, y\}$-graded, the path in bold from a to c has grade x^3y^2x, if we adopt the convention of multiplying elements in $M\{x, y\}$ from left to right as we move along the path. If we switch to treating C as $I\{x, y\}$-graded, the same path has grade $xyx \in I\{x, y\}$. This indicates that the path starts in X, then goes into Y, and then comes back into X.

Another approach to emergence involves understanding how the first homology of the pushout graph $X +_{\mathrm{disc}(B)} Y$ is related to that of the graphs X and Y. For this let us assume that the maps j and k in the pushout diagram



are monic. It follows all the arrows in the diagram are monic, so X and Y are subgraphs of $X +_{\mathrm{disc}(B)} Y$ having no edges in common, and their intersection is the discrete graph $\mathrm{disc}(B)$. This is the case in Example 9.1. To simplify the notation further, let us define

$$X \cup Y = X +_{\mathrm{disc}(B)} Y, \qquad X \cap Y = \mathrm{disc}(B).$$

In this situation, the first homology with coefficients in an abelian group A is well-understood. Here the classical Mayer–Vietoris sequence [22] reduces to the following exact sequence of abelian groups:

$$0 \to H_1(X, A) \oplus H_1(Y, A) \to H_1(X \cup Y, A) \to$$

$$H_0(X \cap Y, A) \to H_0(X, A) \oplus H_0(Y, A) \to H_0(X \cup Y, A) \to 0.$$

The abelian group $H_1(X, A) \oplus H_1(Y, A)$ consists of **non-emergent cycles**, which existed in X and Y before we glued these graphs together along the vertices in B. It is a subgroup of the group of actual interest, $H_1(X \cup Y)$. The quotient of $H_1(X \cup Y)$ by this subgroup may thus be seen as the group of 'emergent' 1-cycles.

We thus focus on this portion of the Mayer–Vietoris exact sequence:

$$0 \to H_1(X, A) \oplus H_1(Y, A) \xrightarrow{\iota} H_1(X \cup Y, A) \xrightarrow{\partial} H_0(X \cap Y, A) \tag{4}$$

and define the group of **emergent 1-cycles** to be

$$\operatorname{coker} \iota = \frac{H_1(X \cup Y, A)}{\operatorname{im} \iota} = \frac{H_1(X \cup Y, A)}{\ker \partial}.$$

There is a short exact sequence expressing the group of 1-cycles on $X \cup Y$ as an extension of the group of emergent 1-cycles by the group of non-emergent 1-cycles:

$$0 \to H_1(X, A) \oplus H_1(Y, A) \xrightarrow{\iota} H_1(X \cup Y, A) \to \operatorname{coker} \iota \to 0.$$

To go further we need explicit formulas for ι and ∂:

- $\iota = \iota_X + \iota_Y$ where $\iota_X \colon H_1(X, A) \to H_1(X \cup Y, A)$ is the map induced from the inclusion $i_X \colon X \to X \cup Y$, and similarly $\iota_Y \colon H_1(Y, A) \to H_1(X \cup Y, A)$ is the map induced from i_Y.
- ∂ may be defined as follows. We can uniquely express $c \in H_1(X \cup Y, A) \subseteq C_1(X \cup Y, A)$ as a sum of 1-chains

  $$c_X \in C_1(X, A) \subseteq C_1(X \cup Y, A), \qquad c_Y \in C_1(Y, A) \subseteq C_1(X \cup Y, A).$$

  These 1-chains are generally not themselves 1-cycles, but their boundaries sum to zero, and the boundary of c_X lies in $C_1(X, A)$ while that of c_Y lies in $C_1(Y, A)$, so

  $$dc_X = -dc_Y \in C_0(X \cap Y, A) \subseteq C_0(X \cup Y, A).$$

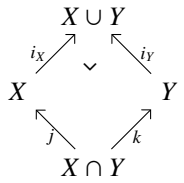
  We thus define ∂c by

  $$\partial c = dc_X = -dc_Y \in C_0(X \cap Y, A) = H_0(X \cap Y, A).$$

Our choice to define ∂c to be dc_X rather than dc_Y is an arbitrary sign convention.

Next we need to adapt these ideas to homology with coefficients in a commutative monoid C. A case of special interest is $C = \mathbb{N}$, since the first homology of a graph with coefficients in $\mathbb{N}$ controls feedback as in Corollary 8.8, and we have a fairly clear picture of this first homology thanks to Theorems 8.6 and 8.12.

We continue to assume we have a union of graphs whose intersection is a discrete graph, i.e., a graph with no edges:



It is notationally convenient to treat $H_1(X,C) \subseteq C_1(X,C), H_1(Y,C) \subseteq C_1(Y,C)$ and $H_1(X \cup Y, C)$ as submonoids of $C_1(X \cup Y, C)$. Since every edge of $X \cup Y$ is either an edge of X or of Y, but not both, we have

$$C_1(X \cup Y, C) = C_1(X,C) \oplus C_1(Y,C)$$

where we write an equals sign because this is an 'internal direct sum': every element $c \in C_1(X \cup Y, C)$ can be uniquely written as a sum of elements $c_X \in C_1(X,C)$ and $c_Y \in C_1(Y,C)$. We define monoid homomorphisms $p_X, p_Y \colon C_1(X \cup Y, C) \to C_1(X \cup Y, C)$ by

$$p_X(c) = c_X, \qquad p_Y(c) = c_Y.$$

It is also convenient to treat $C_0(X,C), C_0(Y,C)$ and $C_0(X \cap Y, C)$ as submonoids of $C_0(X \cup Y, C)$. We abbreviate

$$C[s], C[t] \colon C_1(X \cup Y, C) \to C_0(X \cup Y, C)$$

simply as s and t. We can also use these notations for the maps

$$C[s], C[t] \colon C_1(X) \to C_0(Y)$$

and

$$C[s], C[t] \colon C_1(Y) \to C_0(Y)$$

without confusion, since these are restrictions of the maps s and t defined on all of $C_1(X \cup Y, C)$.

The natural map from the coproduct $X + Y$ to the pushout $X \cup Y$ induces a map on homology

$$\iota \colon H_1(X,C) \oplus H_1(Y,C) \to H_1(X \cup Y, C).$$

This map ι sends any pair (c_X, c_Y) to the sum $c_X + c_Y$. This map is a monomorphism, but perhaps not an isomorphism, since there may be emergent cycles. The following Mayer–Vietoris-like theorem clarifies the situation. The theorem simplifies when C is **cancellative**, meaning that

$$c + e = d + e \implies c = d.$$

Theorem 9.2. *If C is a commutative monoid and X, Y are subgraphs of a graph $X \cup Y$ whose intersection $X \cap Y$ is a discrete graph, then the following is an equalizer diagram in the category of commutative monoids:*

$$H_1(X,C) \oplus H_1(Y,C) \xrightarrow{\ \iota\ } H_1(X \cup Y, C) \underset{(tp_X, tp_Y)}{\overset{(sp_X, sp_Y)}{\rightrightarrows}} C_0(X,C) \oplus C_0(Y,C).$$

If C is cancellative, the following diagram is also an equalizer:

$$H_1(X,C) \oplus H_1(Y,C) \xrightarrow{\ \iota\ } H_1(X \cup Y, C) \underset{tp_X}{\overset{sp_X}{\rightrightarrows}} C_0(X,C).$$

as is the analogous diagram with p_X replaced by p_Y.

Proof. For the first equalizer, first we show that $sp_X = tp_X$ and $sp_Y = tp_Y$ on the image of ι. Any element in the image of ι is of the form $c = c_X + c_Y$ with $c_X \in H_1(X,C)$ and $c_Y \in H_1(Y,C)$. Since $p_X c_X = c_X$ and $p_X c_Y = 0$, we have

$$sp_X c = sc_X = tc_X = tp_X c.$$

Similarly we have $sp_Y c = tp_Y c$.

Next we show that any $c \in H_1(X \cup Y, C)$ with $sp_X c = tp_X c$ and $sp_Y c = tp_Y c$ is in the image of ι. These equations say that $p_X c \in H_1(X,C)$ and $p_Y c \in H_1(Y,C)$, so $c = p_X c + p_Y c = \iota(p_X c, p_Y c)$.

We can reduce the second equalizer diagram to the first if we show $sp_X c = tp_X c$ implies $sp_Y c = tp_Y c$ for any $c \in H_1(X \cup Y, C)$. Here we need C to be cancellative. Since c is a cycle we have $sc = tc$ and thus

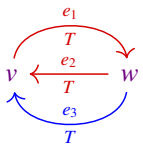
$$sp_X c + sp_Y c = tp_X c + tp_Y c.$$

Since C is cancellative so is $C_0(X \cup Y, C)$, so we can subtract the equation $sp_X c = tp_X c$ from the above equation and conclude

$$sp_Y c = tp_Y c.$$

The diagram with p_X replaced by p_Y works the same way. □

Example 9.3. If C is not cancellative, it is possible to have a cycle $c_X \in H_1(X, C)$ and a 1-chain that is not a cycle, $c_Y \in C_1(Y, C)$, whose sum is a cycle $c_X + c_Y \in H_1(X \cup Y, C)$.



Above is a $\mathbb{B}$-labeled graph where $\mathbb{B} = \{0, 1\}$ with addition as in Example 6.2. The subgraph X contains only the red edges, while the subgraph Y contains only the blue edges. The vertices lie in $X \cap Y$, which is the discrete graph on these two vertices. Let

$$c_X = e_1 + e_2, \qquad c_Y = e_3.$$

Then c_X is a cycle, c_Y is not a cycle, and $c_X + c_Y$ is a cycle because

$$s(c_X + c_Y) = s(e_1 + e_2 + e_3) = v + w + w = v + w, \qquad t(c_X + c_Y) = t(e_1 + e_2 + e_3) = w + v + v = v + w.$$

We can state a Mayer–Vietoris theorem that more closely resembles Equation (4) if we use the monoid homomorphism

$$q \colon C_0(X, C) \to C_0(X \cap Y, C)$$

given by

$$q\Big(\sum_{v \in \{\text{vertices of } X\}} c_v\, v\Big) = \sum_{v \in \{\text{vertices of } X \cap Y\}} c_v\, v.$$

That homomorphism kills off all vertices of X that are not also in Y. A priori $H_0(X \cap Y, C)$ is a quotient of $C_0(X \cap Y, C)$, but $X \cap Y$ has no edges so the quotient map is an isomorphism $C_0(X \cap Y, C) \xrightarrow{\sim} H_0(X \cap Y, C)$. We shall use this isomorphism to identify these two monoids, and treat q as a monoid homomorphism

$$q \colon C_0(X, C) \to H_0(X \cap Y, C).$$

Theorem 9.4. *If C is a cancellative commutative monoid and X, Y are subgraphs of a graph $X \cup Y$ whose intersection $X \cap Y$ is a discrete graph, then the following is an equalizer diagram in the category of commutative monoids:*

$$H_1(X, C) \oplus H_1(Y, C) \xrightarrow{\ \iota\ } H_1(X \cup Y, C) \underset{qtp_X}{\overset{qsp_X}{\rightrightarrows}} H_0(X \cap Y, C).$$

By the symmetry between X and Y, the analogous diagram with p_X replaced by p_Y is also an equalizer.

Proof. By Theorem 9.2 it suffices to show that $c \in H_1(X \cup Y, C)$ has $sp_X c = tp_X c$ if and only if $qsp_X c = qtp_X c$. One direction of the implication is obvious, so we suppose $qsp_X c = qtp_X c$ and aim to show that $sp_X c = tp_X c$.

We write

$$c = \sum_{e \in \{\text{edges of } X \cup Y\}} c_e e.$$

Since c is a cycle we have

$$\sum_{e \in \{\text{edges of } X \cup Y\}} c_e s(e) = \sum_{e \in \{\text{edges of } X \cup Y\}} c_e t(e).$$

There are three mutually exclusive choices for a vertex in $X \cup Y$: it is either 1) in X but not Y, 2) in $X \cap Y$, or 3) Y but not in X. In case 1) we say the vertex is in $X - Y$ and in case 3) we say the vertex is in $Y - X$, merely by way of abbreviation. The above equation thus implies three equations, of which the important one is the first:

$$\sum_{e \in \{\text{edges of } X \cup Y \text{ whose source is in } X - Y\}} c_e s(e) = \sum_{e \in \{\text{edges of } X \cup Y \text{ whose target is in } X - Y\}} c_e t(e)$$

Since an edge of $X \cup Y$ whose source is in $X - Y$ must be an edge of X, this equation is equivalent to

$$\sum_{e \in \{\text{edges of } X \text{ whose source is in } X - Y\}} c_e s(e) = \sum_{e \in \{\text{edges of } X \text{ whose target is in } X - Y\}} c_e t(e).$$

Since $qsp_X c = qtp_X c$ we also know that

$$\sum_{e \in \{\text{edges of } X \text{ whose source is in } X \cap Y\}} c_e s(e) \quad = \quad \sum_{e \in \{\text{edges of } X \text{ whose target is in } X \cap Y\}} c_e t(e).$$

Adding the last two equations we get

$$\sum_{e \in \{\text{edges of } X\}} c_e s(e) = \sum_{e \in \{\text{edges of } X\}} c_e t(e).$$

This says $sp_X c = tp_X c$, as desired. □

10. Conclusion

We hope these results can be used to further the use of causal loop diagrams—that is, $\{+,-\}$-labeled graphs—and other monoid-labeled graphs in system dynamics, systems biology and other related fields. Software already exists that explicitly uses category theory to work with causal loop diagrams and other monoid-labeled graphs. For example, AlgebraicJulia is a software platform for doing scientific computing with categories, and it has a package called StockFlow.jl [5, 47] that allows for the creation and composition of open causal loop diagrams [9]. Redekopp has created a program called ModelCollab, based on StockFlow.jl, whose web-based interface enables collaborative work with causal loop diagrams [39]. Separately, Patterson and collaborators are building a software framework called CatColab, based on double categories, and they have used it to create tools for finding motifs in M-labeled graphs when:

- $M = \{+,-\}$: causal loop diagrams [10].
- $M = \{+,-\} \times \mathbb{B}$: causal loop diagrams with boolean-valued delays [11].
- $M = \{+,-,0\}$: causal loop diagrams with indeterminate effects [12].

We hope that some more ideas from this paper can be embodied in software.

On the other hand, many large databases of labeled graphs have already been compiled, some listed in Pathguide [40]. A visually charismatic example is the KEGG database of pathways for metabolism, biological information processing, and other biological processes [24]. There is thus a great opportunity for applying new mathematics and software to analyze these labeled graphs. In addition, there is an opportunity for better cross-fertilization between systems biology and ideas from system dynamics [46], such as system archetypes [26].

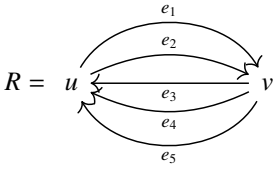
There are also many mathematical questions to study. We briefly mention two. First, in Section 6 we tentatively explored rig-labeled graphs. Given a rig R, it might be interesting to develop a category of R-labeled graphs where several edges 'in series' can be mapped to a single edge with their labels being multiplied:

$$v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \cdots \xrightarrow{r_k} v_k \qquad \mapsto \qquad v_0 \xrightarrow{r_1 \cdots r_k} v_k$$

and also several edges 'in parallel' can be mapped to a single edge with their labels being added:

We have seen that the former pattern shows up in the *opposite* of the category of monoid-labeled graphs and Kleisli morphisms, while the latter shows up in the category of commutative-monoid labeled graphs and additive morphisms.

Second, it would be interesting to understand which commutative monoids arise as $H_1(G, \mathbb{N})$ for some finite graph G. As seen in Example 8.7, not all of them are free. The example there has a presentation with four minimal cycles as generators, one relation, and no syzygies (that is, relations between relations). The following graph



has $H_1(R, \mathbb{N})$ with 6 minimal cycles as generators, 3 relations, and one syzygy. This suggests that it might be interesting to study the homology of the commutative monoids $H_1(G, \mathbb{N})$ for finite graphs G. (For a

simple introduction to the homology of monoids see Lafont [29, Sec. 4].) More generally, it suggests that the homology $H_1(G, C)$ with coefficients in a commutative monoid C is worth studying more deeply.

References

[1] Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. Cambridge U. Press, 1994 (cit. on pp. 10, 11).

[2] Rebekah Aduddell et al. "A compositional account of motifs, mechanisms, and dynamics in biochemical regulatory networks". In: *Compositionality* 6 (2024). DOI: 10.32408/compositionality-6-2. arXiv: 2301.01445 (cit. on pp. 7, 12).

[3] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall, 2006 (cit. on p. 2).

[4] Uri Alon. "Network motifs: theory and experimental approaches". In: *Nature Reviews Genetics* 8.6 (2007), pp. 450–461 (cit. on p. 11).

[5] John Baez et al. "Compositional Modeling with Stock and Flow Diagrams". In: *Electronic Proceedings in Theoretical Computer Science* 380 (Aug. 2023), 77–96. ISSN: 2075-2180. DOI: 10.4204/eptcs.380.5 (cit. on p. 35).

[6] John C. Baez and Kenny Courser. "Structured cospans". In: *Theory Appl. Categ.* 35 (2020), Paper No. 48, 1771–1822. arXiv: 1911.04630 (cit. on pp. 3, 18, 20, 22).

[7] John C. Baez, Kenny Courser, and Christina Vasilakopoulou. "Structured versus decorated cospans". In: *Compositionality* 4.3 (2022), p. 39. DOI: 10.32408/compositionality-4-3. arXiv: 2101.09363 (cit. on pp. 3, 18, 20, 23–25).

[8] John C. Baez and Jade Master. "Open Petri nets". In: *Math. Structures Comput. Sci.* 30.3 (2020), pp. 314–341. DOI: 10.1017/s0960129520000043. arXiv: 1808.05415 (cit. on p. 2).

[9] John C. Baez et al. "A categorical framework for modeling with stock and flow diagrams". In: *Mathematics of Public Health: Mathematical Modelling from the Next Generation*. Ed. by Jummy David and Jianhong Wu. Springer, 2023, pp. 175–207. DOI: https://doi.org/10.1007/978-3-031-40805-2. arXiv: 2211.01290 (cit. on pp. 5, 35).

[10] *CatColab: causal loop diagram*. URL: https://catcolab.org/help/theory/causal-loop (cit. on p. 35).

[11] *CatColab: causal loop diagram with delays*. URL: https://catcolab.org/help/theory/causal-loop-delays (cit. on p. 35).

[12] *CatColab: causal loop diagram with indeterminates*. URL: https://catcolab.org/help/theory/indeterminate-causal-loop (cit. on p. 35).

[13] Adittya Chaudhuri, Ralf Köhl, and Olaf Wolkenhauer. *A mathematical framework to study organising principles in graphical representations of biochemical processes*. 2024. arXiv: 2410.18024 (cit. on p. 2).

[14] Kenny Courser. "Open Systems: A Double Categorical Perspective". PhD thesis. U. C. Riverside, 2020. arXiv: 2008.02394 (cit. on pp. 3, 18).

[15] Eric H. Davidson. *The Regulatory Genome: Gene Regulatory Networks in Development and Evolution*. Elsevier, 2006 (cit. on p. 2).

[16] Brendan Fong. "Decorated cospans". In: *Theory Appl. Categ.* 30 (2015), Paper No. 33, 1096–1120 (cit. on pp. 3, 23).

[17] Brendan Fong. "The Algebra of Open and Interconnected Systems". PhD thesis. Cambridge U., 2016. arXiv: 1609.05382 (cit. on pp. 3, 18, 23).

[18] J. W. Forrester. *Industrial Dynamics*. Pegasus Communications, 1961 (cit. on p. 2).

[19] J. W. Forrester. *Urban Dynamics*. Pegasus Communications, 1969 (cit. on p. 2).

[20] Abraham Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968 (cit. on p. 9).

[21] Jonathan S. Golan. *Semirings and Their Applications*. Springer, 1999 (cit. on p. 18).

[22] Allen Hatcher. *Algebraic Topology*. Cambridge U. Press, 2009 (cit. on pp. 3, 32).

[23] P. S. Hovmand. *Community Based System Dynamics*. Springer, 2014 (cit. on p. 2).

[24] Minoru Kanehisa et al. "KEGG: biological systems database as a model of the real world". In: *Nucleic Acids Research* 53.D1 (2025), pp. D672–D677. DOI: https://doi.org/10.1093/nar/gkae909 (cit. on pp. 2, 35).

[25] Stuart Kauffman. "Homeostasis and differentiation in random genetic control networks". In: *Nature* 224.5215 (1969), pp. 177–178. DOI: https://doi.org/10.1038/224177a0 (cit. on p. 2).

[26] Daniel H. Kim and Virginia Anderson. *Systems Archetype Basics*. Pegasus Communications Inc, 1998 (cit. on p. 35).

[27] Edda Klipp et al. *Systems Biology: a Textbook*. John Wiley & Sons, 2016 (cit. on p. 2).
[28] Anders Kock. "Closed categories generated by commutative monads". In: *Journal of the Australian Mathematical Society* 12.4 (1971), pp. 405–424. DOI: doi:10.1017/S1446788700010272 (cit. on p. 27).
[29] Yves Lafont and A Proute. "Church-Rosser property and homology of monoids". In: *Math. Struct. Comp. Sci.* 1 (3 1991), pp. 297–326. DOI: doi:10.1017/S096012950000133X (cit. on p. 36).
[30] Nicolas Le Novère et al. "The Systems Biology Graphical Notation". In: *Nature Biotechnology* 27.8 (Aug. 2009), pp. 735–741. DOI: 10.1038/nbt.1558 (cit. on p. 2).
[31] Martina Lembani et al. "Understanding key drivers of performance in the provision of maternal health services in eastern cape, South Africa: a systems analysis using group model building". In: *BMC Health Services Research* 18 (2018), pp. 1–12. DOI: https://doi.org/10.1186/ (cit. on p. 1).
[32] Fosco Loregian and Emily Riehl. "Categorical notions of fibration". In: *Expositiones Mathematicae* 38.4 (2020), pp. 496–514. DOI: https://doi.org/10.1016/j.exmath.2019.02.004. arXiv: 1806.0612 (cit. on pp. 5, 6).
[33] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 2012 (cit. on p. 6).
[34] William S. Massey. *Singular Homology Theory*. Springer, 1980 (cit. on p. 27).
[35] Jade Master. "Composing Behaviors of Networks". PhD thesis. U. C. Riverside, 2021. arXiv: 2105.12905 (cit. on p. 17).
[36] Jade Master. *The open algebraic path problem*. 2021. arXiv: 2005.06682 (cit. on p. 17).
[37] José Meseguer and Ugo Montanari. "Petri nets are monoids". In: *Information and Computation* 88.2 (1990), pp. 105–155 (cit. on p. 27).
[38] Huaiyu Mi et al. "Systems Biology Graphical Notation: Activity Flow Language Level 1 Version 1.2". In: *Journal of Integrative Bioinformatics* 12.2 (Sept. 2015), p. 265. DOI: 10.2390/biecoll-jib-2015-265 (cit. on p. 2).
[39] *ModelCollab*. URL: https://modelcollab.usask.ca/ (cit. on p. 35).
[40] *Pathguide: the pathway resource list*. URL: http://pathguide.org/ (cit. on p. 35).
[41] Evan Patterson. "Structured and decorated cospans from the viewpoint of double category theory". In: *Electronic Proceedings in Theoretical Computer Science* 397 (2023), pp. 210–225. DOI: 10.4204/eptcs.397.13. arXiv: 2304.00447 (cit. on pp. 18, 20, 21).
[42] Adrien Rougny et al. "Systems Biology Graphical Notation: Process Description Language Level 1 Version 2.0". In: *Journal of Integrative Bioinformatics* 16.2 (June 2019). DOI: 10.1515/jib-2019-0022 (cit. on p. 2).
[43] Christopher Sooka and Agnes Rwashana-Semwanga. "Modeling the dynamics of maternal healthcare in Uganda: a system dynamics approach". In: *World Journal of Modeling and Simulation* 7 (2011), pp. 163–172 (cit. on p. 1).
[44] Anatoly Sorokin et al. "Systems Biology Graphical Notation: Entity Relationship Language Level 1 Version 2". In: *Journal of Integrative Bioinformatics* 12.2 (Sept. 2015), p. 264. DOI: 10.2390/biecoll-jib-2015-264 (cit. on p. 2).
[45] John D. Sterman. *Business Dynamics*. McGraw-Hill, Inc., 2000 (cit. on p. 2).
[46] John D. Sterman. *System Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw-Hill, 2002 (cit. on pp. 7, 35).
[47] *StockFlow.jl*. URL: https://github.com/AlgebraicJulia/StockFlow.jl (cit. on p. 35).
[48] John J. Tyson and Béla Novák. "Functional motifs in biochemical reaction networks". In: *Annual Review of Physical Chemistry* 61.1 (2010), pp. 219–240 (cit. on p. 12).