MONOIDAL WIDTH

ELENA DI LAVORE ● AND PAWEŁ SOBOCIŃSKI ●

Tallinn University of Technology, Tallinn, Estonia

ABSTRACT. We introduce monoidal width as a measure of complexity for morphisms in monoidal categories. Inspired by well-known structural width measures for graphs, like tree width and rank width, monoidal width is based on a notion of syntactic decomposition: a monoidal decomposition of a morphism is an expression in the language of monoidal categories, where operations are monoidal products and compositions, that specifies this morphism. Monoidal width penalises the composition operation along "big" objects, while it encourages the use of monoidal products. We show that, by choosing the correct categorical algebra for decomposing graphs, we can capture tree width and rank width. For matrices, monoidal width is related to the rank. These examples suggest monoidal width as a good measure for structural complexity of processes modelled as morphisms in monoidal categories.

CONTENTS

1. Introduction

In recent years, a current of research has emerged with focus on the interaction of *structure* — especially algebraic, using category theory and related subjects — and *power*, that is algorithmic and combinatorial insights stemming from graph theory, game theory and related subjects. Recent works include [ADW17, AS21, MS22].

The algebra of monoidal categories is a fruitful source of *structure* — it can be seen as a general process algebra of concurrent processes, featuring a sequential (;) as well as a parallel ($\otimes$) composition. Serving as a process algebra in this sense, it has been used to describe artefacts of a computational nature as *arrows* of appropriate monoidal categories. Examples include Petri nets [FS18], quantum circuits [CK17, DKPvdW20], signal flow graphs [FS18, BSZ21], electrical circuits [CK22, BS21], digital circuits [GJL17], stochastic processes [Fri20, CJ19] and games [GHWZ18].

Given that the algebra of monoidal categories has proved its utility as a language for describing computational artefacts in various applications areas, a natural question is to examine its relationship with *power*: can monoidal structure help us to design efficient algorithms? To begin to answer this question, let us consider a mainstay of computer science: *divide-and-conquer* algorithms. Such algorithms rely on the internal geometry of the global artefact under consideration to ensure the ability to *divide*, that is, decompose it consistently into simpler components, inductively compute partial solutions on the components, and then recombine these local results to obtain a global solution.

Figure 1. This morphism can be decomposed in two different ways: $(f \otimes f')\,;(g \otimes g') = (f\,;g) \otimes (f'\,;g')$.

Let us now return to systems described as arrows of monoidal categories. In applications, the parallel ($\otimes$) composition typically means placing systems side-by-side with no explicit interconnections. On the other hand, the sequential (;) composition along an object typically means communication, resource sharing or synchronisation, the complexity of which is determined by the object along which the composition is performed. Based on examples in the literature, our basic motivating intuition is:

> *An algorithmic problem on an artefact that is a '$\otimes$' lends itself to a divide-and-conquer approach more easily than one that is a ';'.*

Moreover, the "size" of the object along which the ';' occurs matters; typically the "larger" the object, the more work is needed in order to recombine results in any kind of divide-and-conquer approach. An example is compositional reachability checking in Petri nets of

Rathke et. al. [RSS14]: calculating the sequential composition is exponential in the size of the boundary. Another recent example is the work of Master [Mas22] on a compositional approach to calculating shortest paths.

On the other hand, (monoidal) category theory equates different descriptions of systems. Consider what is known as middle-four interchange, illustrated in Figure 1. Although monoidal category theory asserts that $(f \otimes f') \, ; (g \otimes g') = (f \, ; g) \otimes (f' \, ; g')$, considering the two sides of the equations as decomposition blueprints for a divide-and-conquer approach, the right-hand side of the equation is clearly preferable since it maximises parallelism by minimising the size of the boundary along which composition occurs. This, roughly speaking, is the idea of *width* – expressions in the language of monoidal categories are assigned a natural number that measures "how good" they are as decomposition blueprints. The *monoidal width* of an arrow is then the width of its most efficient decomposition. In concrete examples, arrows with low width lend themselves to efficient divide-and-conquer approaches, following a width-optimal expression as a decomposition blueprint.

The study of efficient decompositions of combinatorial artefacts is well-established, especially in graph theory. A number of *graph widths* — by which we refer to related concepts like tree width, path with, branch width, cut width, rank width or twin width — have become known in computer science because of their relationship with algorithmic properties. All of them share a similar basic idea: in each case, a specific notion of legal decomposition is priced according to the most expensive operation involved, and the price of the cheapest decomposition is the width.

Perhaps the most famous of these is tree width, a measure of complexity for graphs that was independently defined by different authors [BB73, Hal76, RS86]. Every nonempty graph has a tree width, which is a natural number. Intuitively, a tree decomposition is a recipe for decomposing a graph into smaller subgraphs that form a tree shape. These subgraphs, when some of their vertices are identified, need to compose into the original graph, as shown in Figure 2. Courcelle's theorem

> *Every property expressible in the monadic second order logic of graphs can be verified in linear time on graphs with bounded tree width.*

is probably the best known among several results that establish links with algorithms [Bod92, BK08, Cou90] thus illustrating its importance for computer science.

FIGURE 2. A tree decomposition cuts the graph along its vertices.

Another important measure is rank width [OS06] — a relatively recent development that has attracted significant attention in the graph theory community. A rank decomposition is a recipe for decomposing a graph into its single-vertex subgraphs by cutting along edges. The cost of a cut is the rank of the adjacency matrix that represents it, as illustrated in Figure 3. An intuition for rank width is that it is a kind of "Kolmogorov complexity" for

graphs, with higher rank widths indicating that the connectivity data of the graph cannot be easily compressed. For example, while the family of cliques has unbounded tree width, their connectivity rather simple: in fact, all cliques have rank width 1.

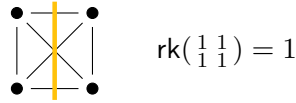$$\mathsf{rk}(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}) = 1$$

Figure 3. A cut and its matrix in a rank decomposition.

Contribution. Building on our conference paper [DLS22], our goals are twofold. Firstly, to introduce the concept of monoidal width and begin to develop techniques for reasoning about it.

Before describing concrete, technical contributions, let us take a bird's eye view. It is natural for the seasoned researcher to be sceptical of a new abstract framework that seeks to generalise known results. The best abstract approaches *(i)* simplify existing known arguments, *(ii)* clean up the research landscape by connecting existing notions, or *(iii)* introduce techniques that allow one to prove new theorems. This paper does not (yet) bring strong arguments in favour of monoidal width if one uses these three points as yardsticks. Our high-level, conceptual contribution is, instead, the fact that the algebra of monoidal categories – already used in several contexts in theoretical computer science – is a multi-purpose algebra for specifying decompositions of graph-like structures important for computer scientists. There are several ways of making this work, and making these monoidal *algebras* of "open graphs" explicit as *monoidal categories* is itself a valuable endeavour. Indeed, identifying a monoidal category automatically yields a particular notion of decomposition: the instance of monoidal width in the monoidal category of interest. This point of view therefore demystifies ad hoc notions of decomposition that accompany each notion of width that we consider in this paper. Moreover, having an explicit algebra is also useful because it suggests a data structure — the expression in the language of monoidal categories — as a way of describing decompositions.

The results in this paper can be seen as a "sanity check" of these general claims, but can also be seen as taking the first technical steps in order to build towards points *(i)-(iii)* of the previous paragraph. To this end we examine monoidal width in the presence of common structure, such as coherent comultiplication on objects, and in a foundational setting such as the monoidal category of matrices. Secondly, connecting this approach with previous work, to examine graph widths through the prism of monoidal width. The two widths we focus on are tree width and rank width. We show that both can be seen as instances of monoidal width. The interesting part of this endeavour is identifying the monoidal category, and thus the relevant "decomposition algebra" of interest.

Unlike the situation with graph widths, it does not make sense to talk about monoidal width per se, since it is dependent on the choice of underlying monoidal category and thus a particular "decomposition algebra". The decomposition algebras that underlie tree and rank decompositions reflect their intuitive understanding. For tree width, this is a cospan category whose morphisms represent graphs with vertex interfaces, while for rank width it is a category whose morphisms represent graphs with edge interfaces, with adjacency matrices playing the role of tracking connectivity information within a graph. We show that the

monoidal width of a morphism in these two categories is bounded, respectively, by the branch (Theorem 3.34) and rank width (Theorem 5.26) of the corresponding graph. In the first instance, this is enough to establish the connection between monoidal width and tree width, given that it is known that tree width and branch width are closely related. A small technical innovation is the definition of intermediate inductive notions of branch (Definition 3.14) and rank (Definition 5.7) decompositions, equivalent to the original definitions via "global" combinatorial notions of graph decomposition. The inductive presentations are closer in spirit to the inductive definition of monoidal decomposition, and allow us to give direct proofs of the main correspondences.

String diagrams. String diagrams [JS91] are a convenient syntax for monoidal categories, where a morphism $f\colon X \to Y$ is depicted as a box with input and output wires: $X -\boxed{f}- Y$. Morphisms in monoidal categories can be composed sequentially, using the composition of the category, and in parallel, using the monoidal structure. These two kinds of composition are reflected in the string diagrammatic syntax: the sequential composition $f \mathbin{;} g$ is depicted by connecting the output wire of f with the input wire of g; the parallel composition $f \otimes f'$ is depicted by writing f on top of f'.

$$f \mathbin{;} g = \quad X -\boxed{f}-\boxed{g}- Z \qquad\qquad\qquad f \otimes f' = \quad \begin{array}{l} X -\boxed{f}- Y \\ X' -\boxed{f'}- Y' \end{array}$$

The advantage of this syntax is that all coherence equations for monoidal categories are trivially true when written with string diagrams. An example is the middle-four interchange law $(f \otimes f') \mathbin{;} (g \otimes g') = (f \mathbin{;} g) \otimes (g \mathbin{;} g')$. These two expressions have one representation in terms of string diagrams, as shown in Figure 1. The coherence theorem for monoidal categories [Mac78] ensures that string diagrams are a sound and complete syntax for morphisms in monoidal categories.

Related work. This paper contains the results of [DLS21] and [DLS22] with detailed proofs. We generalise the results of [DLS21] to undirected hypergraphs and provide a syntactic presentation of the subcategory of the monoidal category of cospans of hypergraphs on discrete objects.

Previous syntactical approaches to graph widths are the work of Pudlák, Rödl and Savický [PRS88] and the work of Bauderon and Courcelle [BC87]. Their works consider different notions of graph decompositions, which lead to different notions of graph complexity. In particular, in [BC87], the cost of a decomposition is measured by counting *shared names*, which is clearly closely related to penalising sequential composition as in monoidal width. Nevertheless, these approaches are specific to particular, concrete notions of graphs, whereas our work concerns the more general algebraic framework of monoidal categories.

Abstract approaches to width have received some attention recently, with a number of diverse contributions. Blume et. al. [BBFK11], similarly to our work, use (the category of) cospans of graphs as a formal setting to study graph decompositions: indeed, a major insight of loc. cit. is that tree decompositions are tree-shaped diagrams in the cospan category, and the original graph is reconstructed as a colimit of such a diagram. Our approach is more general, however, emphasising the relevance of the algebra of monoidal categories, of which cospan categories are just one family of examples.

The literature on comonads for game semantics characterises tree and path decompositions of relational structures (and graphs in particular) as coalgebras of certain comonads [ADW17, AS21, MS22, AM21, CD21]. Bumpus and Kocsis [BK21, Bum21] and, later, Bumpus, Kocsis and Master [BKM23] also generalise tree width to the categorical setting, although their approach is conceptually and technically removed from ours. Their work takes a combinatorial perspective on decompositions, following the classical graph theory literature. Given a shape of decomposition, called the spine in [BK21], a decomposition is defined globally as a functor out of that shape. This generalises the characterisation of tree width based on Halin's S-functions [Hal76]. In contrast, monoidal width is algebraic in flavour, following Bauderon and Courcelle's insights on tree decompositions [BC87]. Monoidal decompositions are syntax trees defined inductively and rely on the decomposition algebra given by monoidal categories.

Synopsis. The definition of monoidal width is introduced in Section 2, together with a worked out example. In Section 3 we recover tree width by instantiating monoidal width in a suitable category of cospans of hypergraphs. We recall it in Section 3.3 and provide a syntax for it in Section 3.4. Similarly, in Section 5 we recover rank width by instantiating monoidal width in a prop of graphs with boundaries where the connectivity information is stored in adjacency matrices, which we recall in Section 5.3. This motivates us to study monoidal width for matrices over the natural numbers in Section 4.

2. Monoidal width

We introduce monoidal width, a notion of complexity for morphisms in monoidal categories that relies on explicit syntactic *decompositions*, relying on the algebra of monoidal categories. We then proceed with a simple, yet useful examples of efficient monoidal decompositions in Section 2.1.

A monoidal decomposition of a morphism f is a binary tree where internal nodes are labelled with the operations of composition ; or monoidal product $\otimes$, and leaves are labelled with "atomic" morphisms. A decomposition, when evaluated in the obvious sense, results in f. We do not assume that the set of atomic morphisms $\mathcal{A}$ is minimal, they are merely morphisms that do not necessarily need to be further decomposed. We assume that $\mathcal{A}$ contains enough atoms to have a decomposition for every morphism. In most cases, we will take $\mathcal{A}$ to contain all the morphisms.

Definition 2.1 (Monoidal decomposition). Let C be a monoidal category and $\mathcal{A}$ be a subset of its morphisms to which we refer as *atomic*. The set D_f of monoidal decompositions of $f \colon A \to B$ in C is defined inductively:

$$
\begin{aligned}
D_f \quad ::= \quad & (f) & & \text{if } f \in \mathcal{A} \\
\mid \quad & (d_1 \text{---} \otimes \text{---} d_2) & & \text{if } d_1 \in D_{f_1}, d_2 \in D_{f_2} \text{ and } f =_{\mathsf{C}} f_1 \otimes f_2 \\
\mid \quad & (d_1 \text{---} ;_X \text{---} d_2) & & \text{if } d_1 \in D_{f_1 \colon A \to X}, d_2 \in D_{f_2 \colon X \to B} \text{ and } f =_{\mathsf{C}} f_1 ; f_2
\end{aligned}
$$

In general, a morphism can be decomposed in different ways and decompositions that maximise parallelism are deemed more efficient. The monoidal width of a morphism is the cost of its cheapest monoidal decomposition.

Formally, each operation and atom in a decomposition is assigned a weight that will determine the cost of the decomposition. This is captured by the concept of a *weight function*.

Definition 2.2. Let C be a monoidal category and let $\mathcal{A}$ be its atomic morphisms. A *weight function* for $(\mathsf{C}, \mathcal{A})$ is a function $\mathsf{w}\colon \mathcal{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{C}) \to \mathbb{N}$ such that $\mathsf{w}(X \otimes Y) = \mathsf{w}(X) + \mathsf{w}(Y)$, and $\mathsf{w}(\otimes) = 0$.

A prop is a strict symmetric monoidal category where objects are natural numbers and the monoidal product on them is addition. If C is a prop, then, typically, we let $\mathsf{w}(1) := 1$. The idea behind giving a weight to an object $X \in \mathsf{C}$ is that $\mathsf{w}(X)$ is the cost paid for composing along X.
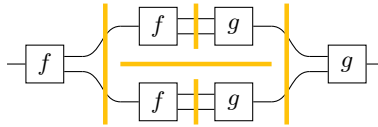
Definition 2.3 (Monoidal width)**.** Let w be a weight function for $(\mathsf{C}, \mathcal{A})$. Let f be in C and $d \in D_f$. The *width* of d is defined inductively as follows:

$$\mathsf{wd}(d) := \mathsf{w}(f) \qquad\qquad\qquad\qquad \text{if } d = (f)$$
$$\max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2)\} \qquad\qquad \text{if } d = (d_1 - \otimes - d_2)$$
$$\max\{\mathsf{wd}(d_1),\ \mathsf{w}(X),\ \mathsf{wd}(d_2)\} \qquad \text{if } d = (d_1 - ;_X - d_2)$$

The *monoidal width* of f is $\mathsf{mwd}(f) := \min_{d \in D_f} \mathsf{wd}(d)$.

Example 2.4. Let $f\colon 1 \to 2$ and $g\colon 2 \to 1$ be morphisms in a prop such that $\mathsf{mwd}(f) = \mathsf{mwd}(g) = 2$. The following figure represents the monoidal decomposition of $f;(f \otimes f);(g \otimes g);g$ given by
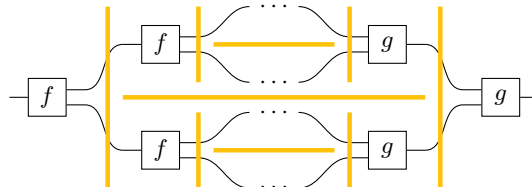
$$(f - ;_2 - (((f - ;_2 - g) - \otimes - (f - ;_2 - g)) - ;_2 - g)).$$

Indeed, taking advantage of string diagrammatic syntax, decompositions can be illustrated by enhancing string diagrams with additional annotations that indicate the order of decomposition. Throughout this paper, we use thick yellow dividing lines for this purpose.

Given that the width of a decomposition is the most expensive operation or atom, the above has width is 2 as compositions are along at most 2 wires.

Example 2.5. With the data of Example 2.4, define a family of morphisms $h_n\colon 1 \to 1$ inductively as $h_0 := f ;_2 g$, and $h_{n+1} := f ;_2 (h_n \otimes h_n) ;_2 g$.
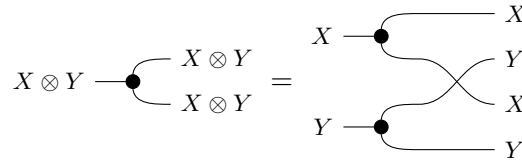

Each h_n has a decomposition of width 2^n where the root node is the composition along the middle wires. However — following the schematic diagram above — we have that $\mathsf{mwd}(h_n) \le 2$ for any n.

2.1. **Monoidal width of copy.** Although monoidal width is a very simple notion, reasoning about it in concrete examples can be daunting because of the combinatorial explosion in the number of possible decompositions of any morphism. For this reason, it is useful to examine some commonly occurring structures that one encounters "in the wild" and examine their decompositions. One such situation is when the objects are equipped with a coherent comultiplication structure.

Definition 2.6. Let C be a symmetric monoidal category, with symmetries $\times_{X,Y} \colon X \otimes Y \to Y \otimes X$. We say that C has coherent copying if there is a class of objects $\Delta_\mathsf{C} \subseteq \mathsf{Obj}(\mathsf{C})$, satisfying
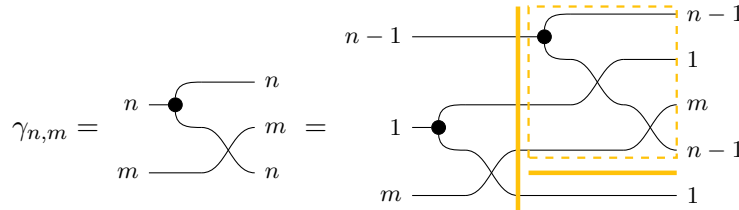
- $X, Y \in \Delta_\mathsf{C}$ iff $X \otimes Y \in \Delta_\mathsf{C}$;
- Every object $X \in \Delta_\mathsf{C}$ is endowed with a morphism $\multimap_X \colon X \to X \otimes X$;
- For every $X, Y \in \Delta_\mathsf{C}$, $\multimap_{X\otimes Y} = (\multimap_X \otimes \multimap_Y) \,;(\mathbb{1}_X \otimes \times_{X,Y} \otimes \mathbb{1}_Y)$ (coherence).

An example is any cartesian prop, where the copy morphisms are the universal ones given by the cartesian structure: $\multimap_n := \langle \mathbb{1}_n, \mathbb{1}_n \rangle \colon n \to n + n$. For props with coherent copy, we assume that copy morphisms, symmetries and identities are atoms, $\multimap_X, \times_{X,Y}, \mathbb{1}_X \in \mathcal{A}$, and that their weight is given by $\mathsf{w}(\multimap_X) := 2 \cdot \mathsf{w}(X)$, $\mathsf{w}(\times_{X,Y}) := \mathsf{w}(X) + \mathsf{w}(Y)$ and $\mathsf{w}(\mathbb{1}_X) := \mathsf{w}(X)$.

Example 2.7. Let C be a prop with coherent copy and suppose that $1 \in \Delta_\mathsf{C}$. This implies that every $n \in \Delta_\mathsf{C}$ and there are copy morphisms $\multimap_n \colon n \to 2n$ for all n. Let $\gamma_{n,m} := (\multimap_n \otimes \mathbb{1}_m) \,;(\mathbb{1}_n \otimes \times_{n,m}) \colon n + m \to n + m + n$. We can decompose $\gamma_{n,m}$ in terms of $\gamma_{n-1,m+1}$ (in the dashed box), $\multimap_1$ and $\times_{1,1}$ by cutting along at most $n + 1 + m$ wires:

$$\gamma_{n,m} = (\mathbb{1}_{n-1} \otimes ((\multimap_1 \otimes \mathbb{1}_1) \,;(\mathbb{1}_1 \otimes \times_{1,1}))) \,;_{n+1+m} (g_{n-1,m+1} \otimes \mathbb{1}_1).$$

This allows us to decompose $\multimap_n = \gamma_{n,0}$ cutting along only $n + 1$ wires. In particular, this means that $\mathsf{mwd}(\multimap_n) \leq n + 1$.

The following lemma generalises the above example and is used in the proofs of some results in later sections, Proposition 3.30 and Proposition 4.6.

Lemma 2.8. *Let C be a symmetric monoidal category with coherent copying. Suppose that $\mathcal{A}$ contains $\multimap_X$ for all $X \in \Delta_\mathsf{C}$, and $\times_{X,Y}$ and $\mathbb{1}_X$ for all $X \in \mathsf{Obj}(\mathsf{C})$. Let $\overline{X} := X_1 \otimes \cdots \otimes X_n$, with $X_i \in \Delta_\mathsf{C}$, $f \colon Y \otimes \overline{X} \otimes Z \to W$ and let $d \in D_f$. Let $\gamma_{\overline{X}}(f) :=$*

$(\mathbb{1}_Y \otimes -\!\!\!\bullet_{\overline{X}} \otimes \mathbb{1}_Z) \,;\, (\mathbb{1}_{Y \otimes \overline{X}} \otimes \times_{\overline{X},Z}) \,;\, (f \otimes \mathbb{1}_{\overline{X}}).$

Then there is a monoidal decomposition $\mathcal{C}_{\overline{X}}(d)$ of $\gamma_{\overline{X}}(f)$ such that
$$\mathsf{wd}(\mathcal{C}_{\overline{X}}(d)) \leq \max\{\mathsf{wd}(d), \mathsf{w}(Y) + \mathsf{w}(Z) + (n+1) \cdot \max_{i=1,\ldots,n} \mathsf{w}(X_i)\}.$$

Proof. Proceed by induction on the number n of objects being copied. If $n = 0$, then we are done because we keep the decomposition d and define $\mathcal{C}_I(d) := d$.

Suppose that the statement is true for any $f' \colon Y \otimes \overline{X} \otimes Z' \to W$. Let $f \colon Y \otimes \overline{X} \otimes X_{n+1} \otimes Z \to W$. By coherence of $-\!\!\!\bullet$, we can rewrite $\gamma_{\overline{X} \otimes X_{n+1}}(f)$.

Let $\gamma_{\overline{X}}(f)$ be the morphism in the above dashed box. By the induction hypothesis, there is a monoidal decomposition $\mathcal{C}_{\overline{X}}(d)$ of $\gamma_{\overline{X}}(f)$ of bounded width: $\mathsf{wd}(\mathcal{C}_{\overline{X}}(d)) \leq \max\{\mathsf{wd}(d), \mathsf{w}(Y) + \mathsf{w}(X_{n+1} \otimes Z) + (n+1) \cdot \max_{i=1,\ldots,n} \mathsf{w}(X_i)\}$. We can use this decomposition to define a monoidal decomposition $\mathcal{C}_{\overline{X} \otimes X_{n+1}}(d)$ of $\gamma_{\overline{X} \otimes X_{n+1}}(f)$ as shown below.

Note that the only cut that matters is the longest vertical one, the composition node along $Y \otimes \overline{X} \otimes X_{n+1} \otimes Z \otimes X_{n+1}$, because all the other cuts are cheaper. The cost of this cut is $\mathsf{w}(Y) + \mathsf{w}(Z) + 2 \cdot \mathsf{w}(X_{n+1}) + \mathsf{w}(\overline{X}) = \mathsf{w}(Y) + \mathsf{w}(Z) + \mathsf{w}(X_{n+1}) + \sum_{i=1}^{n+1} \mathsf{w}(X_i)$. With this observation and applying the induction hypothesis, we can compute the width of the decomposition $\mathcal{C}_{\overline{X} \otimes X_{n+1}}(d)$.

$$\mathsf{wd}(\mathcal{C}_{\overline{X} \otimes X_{n+1}}(d))$$
$$= \max\left\{\mathsf{w}(Y) + \mathsf{w}(Z) + \mathsf{w}(X_{n+1}) + \sum_{i=1}^{n+1} \mathsf{w}(X_i), \mathsf{wd}(\mathcal{C}_{\overline{X}}(d))\right\}$$
$$\leq \max\left\{\mathsf{w}(Y) + \mathsf{w}(Z) + (n+2) \cdot \max_{i=1,\ldots,n+1} \mathsf{w}(X_i), \mathsf{wd}(d),\right.$$
$$\left. \mathsf{w}(Y) + \mathsf{w}(X_{n+1} \otimes Z) + (n+1) \cdot \max_{i=1,\ldots,n} \mathsf{w}(X_i)\right\}$$

$$= \max \left\{ \mathsf{w}(Y) + \mathsf{w}(Z) + (n+2) \cdot \max_{i=1,\dots,n+1} \mathsf{w}(X_i), \mathsf{wd}(d) \right\} \qquad \square$$

3. A monoidal algebra for tree width

Our first case study is tree width of undirected hypergraphs. We show that monoidal width in a suitable monoidal category of hypergraphs is within constant factors of tree width. We rely on branch width, a measure equivalent to tree width, to relate the latter with monoidal width.

After recalling tree and branch width and the bounds between them in Section 3.1, we define the intermediate notion of inductive branch decomposition in Section 3.2 and show its equivalence to that of branch decomposition. Separating this intermediate step allows a clearer presentation of the correspondence between branch decompositions and monoidal decompositions. Section 3.3 recalls the categorical algebra of cospans of hypergraphs and Section 3.4 introduces a syntactic presentations of them. Finally, Section 3.5 contains the main result of the present section, which relates inductive branch decompositions, and thus tree decompositions, with monoidal decompositions.

Classically, tree and branch widths have been defined for finite undirected multihypergraphs, which we simply call hypergraphs. These have undirected edges that connect sets of vertices and they may have parallel edges.

Definition 3.1. A *(multi)hypergraph* $G = (V, E)$ is given by a finite set of vertices V, a finite set of edges E and an adjacency function $\mathsf{ends}\colon E \to \wp(V)$, where $\wp(V)$ indicates the set of subsets of V. A *subhypergraph* of G is a hypergraph $G' = (V', E')$ such that $V' \subseteq V$, $E' \subseteq E$ and $\mathsf{ends}'(e) = \mathsf{ends}(e)$ for all $e \in E'$.

Definition 3.2. Given two hypergraphs $G = (V, E)$ and $H = (W, F)$, a *hypergraph homomorphism* $\alpha\colon G \to H$ is given by a pair of functions $\alpha_V\colon V \to W$ and $\alpha_E\colon E \to F$ such that, for all edges $e \in E$, $\mathsf{ends}_H(\alpha_E(e)) = \alpha_V(\mathsf{ends}_G(e))$.

$$
\begin{array}{ccc}
E & \xrightarrow{\;f_E\;} & F \\
{\scriptstyle \mathsf{ends}_G}\big\downarrow & & \big\downarrow{\scriptstyle \mathsf{ends}_H} \\
\wp(V) & \xrightarrow{\;\wp(f_V)\;} & \wp(W)
\end{array}
$$

Hypergraphs and hypergraph homomorphisms form a category $\mathsf{UHGraph}$, where composition and identities are given by component-wise composition and identities.

Note that the category $\mathsf{UHGraph}$ is not the functor category $[\{\bullet \to \bullet\}, \mathsf{kl}(\wp)]$: their objects coincide but the morphisms are different.

Definition 3.3. The *hyperedge size* of a hypergraph G is defined as $\gamma(G) := \max_{e \in \mathsf{edges}(G)} |\mathsf{ends}(e)|$. A *graph* is a hypergraph with hyperedge size 2.

Definition 3.4. A *neighbour* of a vertex v is a vertex w distinct from v with an edge e such that $v, w \in \mathsf{ends}(e)$. A *path* in a hypergraph is a sequence of vertices $(v_1, \dots, v_n)$ such that, for every $i = 1, \dots, n-1$, v_i and v_{i+1} are neighbours. A *cycle* in a hypergraph is a path where the first vertex v_1 coincides with the last vertex v_n. A hypergraph is *connected* if there is a path between every two vertices. A *tree* is a connected acyclic hypergraph. A tree is *subcubic* if every vertex has at most three neighbours.

Definition 3.5. The set of *binary trees* with labels in a set Λ is either: a leaf (λ) with label $\lambda \in \Lambda$; or a label $\lambda \in \Lambda$ with two binary trees T_1 and T_2 with labels in Λ, $(T_1 - \lambda - T_2)$.

3.1. **Background: tree width and branch width.** Intuitively, tree width measures "how far" a hypergraph G is from being a tree: a hypergraph is a tree iff it has tree width 1. Hypergraphs with tree widths larger than 1 are not trees; for example, the family of cliques has unbounded tree width.
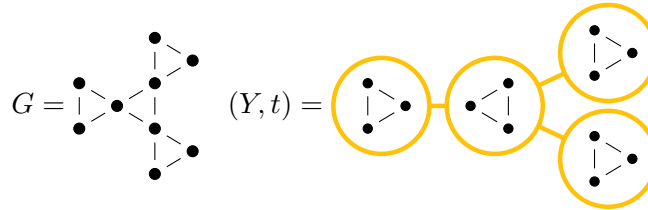
The definition relies on the concept of a *tree decomposition*. For Robertson and Seymour [RS86], a decomposition is itself a tree Y, each vertex of which is associated with a subhypergraph of G. Then G can be reconstructed from Y by identifying some vertices.

Definition 3.6 [RS86]**.** A *tree decomposition* of a hypergraph $G = (V, E)$ is a pair (Y, t) where Y is a tree and $t \colon \mathsf{vertices}(Y) \to \wp(V)$ is a function such that:

(1) Every vertex is in one of the components: $\bigcup_{i \in \mathsf{vertices}(Y)} t(i) = V$.
(2) Every edge has its endpoints in a component: $\forall e \in E \; \exists i \in \mathsf{vertices}(Y) \; \mathsf{ends}(e) \subseteq t(i)$.
(3) The components are glued in a tree shape: $\forall i, j, k \in \mathsf{vertices}(Y) \; i \rightsquigarrow j \rightsquigarrow k \Rightarrow t(i) \cap t(k) \subseteq t(j)$.

The cost is the maximum number of vertices of the component subhypergraphs.

Example 3.7. Consider the hypergraph G and its tree decomposition (Y, t) below. Its cost is 3 as its biggest component has three vertices.

Definition 3.8 (Tree width)**.** Given a tree decomposition (Y, t) of a hypergraph G, its width is $\mathsf{wd}(Y, t) := \max_{i \in \mathsf{vertices}(Y)} |t(i)|$. The tree width of G is given by the min-max formula:

$$\mathsf{twd}(G) := \min_{(Y,t)} \mathsf{wd}(Y, t).$$

Note that Robertson and Seymour subtract 1 from $\mathsf{twd}(G)$ so that trees have tree width 1. To minimise bureaucratic overhead, we ignore this convention.

We use branch width [RS91] as a technical stepping stone to relate monoidal width and tree width. Before presenting its definition, it is important to note that branch width and tree width are *equivalent*, i.e. they are within a constant factor of each other.

Theorem 3.9 [RS91, Theorem 5.1]**.** *Branch width is equivalent to tree width. More precisely, for a hypergraph $G = (V, E)$,*

$$\max\{\mathsf{bwd}(G), \gamma(G)\} \leq \mathsf{twd}(G) \leq \max\{\frac{3}{2}\mathsf{bwd}(G), \gamma(G), 1\}.$$

Branch width relies on branch decompositions, which, intuitively, record in a tree a way of iteratively partitioning the edges of a hypergraph.

Definition 3.10 [RS91]**.** A *branch decomposition* of a hypergraph $G = (V, E)$ is a pair (Y, b) where Y is a subcubic tree and $b \colon \mathsf{leaves}(Y) \cong E$ is a bijection.
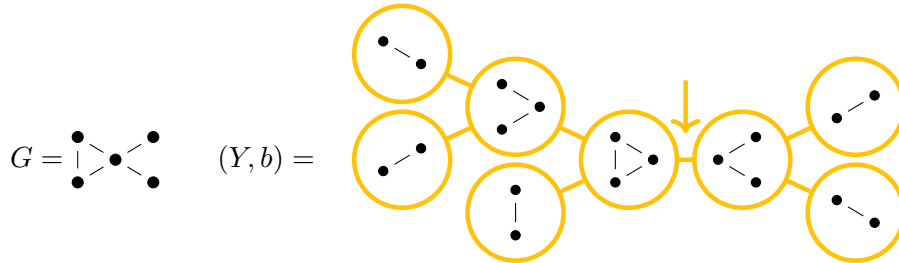
Each edge e in the tree Y determines a splitting of the hypergraph. More precisely, it determines a two partition of the leaves of Y, which, through b, determines a 2-partition $\{A_e, B_e\}$ of the edges of G. This corresponds to a splitting of the hypergraph G into two subhypergraphs G_1 and G_2. Intuitively, the order of an edge e is the number of vertices that are glued together when joining G_1 and G_2 to get G. Given the partition $\{A_e, B_e\}$ of the edges of G, we say that a vertex v of G separates A_e and B_e whenever there are an edge in A_e and an edge in B_e that are both adjacent to v.

Let (Y, b) be a branch decomposition of a hypergraph G. Let e be an edge of Y. The *order* of e is the number of vertices that separate A_e and B_e: $\mathsf{ord}(e) \coloneqq |\mathsf{ends}(A_e) \cap \mathsf{ends}(B_e)|$.

Definition 3.11 (Branch width)**.** Given a branch decomposition (Y, b) of a hypergraph $G = (V, E)$, define its width as $\mathsf{wd}(Y, b) \coloneqq \max_{e \in \mathsf{edges}(Y)} \mathsf{ord}(e)$.

The branch width of G is given by the min-max formula: $\mathsf{bwd}(G) \coloneqq \min_{(Y, b)} \mathsf{wd}(Y, b)$.

Example 3.12. If we start reading the decomposition from an edge in the tree Y, we can extend the labelling to internal vertices by labelling them with the glueing of the labels of their children.

In this example, there is only one vertex separating the first two subgraphs of the decomposition. This means that the corresponding edge in the decomposition tree has order 1.

3.2. **Hypergraphs with sources and inductive definition.** We introduce a definition of decomposition that is intermediate between a branch decomposition and a monoidal decomposition. It adds to branch decompositions the algebraic flavour of monoidal decompositions by using an inductive data type, that of binary trees, to encode a decomposition.

Our approach follows closely Bauderon and Courcelle's hypergraphs with sources [BC87] and the corresponding inductive definition of tree decompositions [Cou92]. Courcelle's result [Cou92, Theorem 2.2] is technically involved as it translates between a combinatorial description of a decomposition to a syntactic one. Our results in this and the next sections are similarly technically involved.

We recall the definition of hypergraphs with sources and introduce inductive branch decompositions of them. Intuitively, the sources of a hypergraph are marked vertices that are allowed to be "glued" together with the sources of another hypergraph. Thus, the equivalence between branch decompositions and inductive branch decompositions formalises the intuition that a branch decomposition encodes a way of dividing a hypergraph into smaller subgraphs by "cutting" along some vertices.

Definition 3.13 [BC87]**.** A *hypergraph with sources* is a pair $\Gamma = (G, X)$ where $G = (V, E)$ is a hypergraph and $X \subseteq V$ is a subset of its vertices, called the sources (Figure 4). Given two hypergraphs with sources $\Gamma = (G, X)$ and $\Gamma' = (G', X')$, we say that Γ' is a subhypergraph of Γ whenever G' is a subhypergraph of G.

Note that the sources of a subhypergraph Γ' of Γ need not to appear as sources of Γ, nor vice versa. In fact, if Γ is obtained by identifying all the sources of Γ_1 with some of the sources of Γ_2, the sources of Γ and Γ_1 will be disjoint.
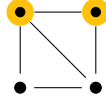

FIGURE 4. Sources are marked vertices in the graph and are thought of as an interface that can be glued with that of another graph.

An inductive branch decomposition is a binary tree whose vertices carry subhypergraphs Γ' of the ambient hypergraph Γ. This set of all such binary trees is defined as follows

$$T_\Gamma \; ::= \; () \; \mid \; (T_\Gamma, \Gamma', T_\Gamma)$$

where Γ' ranges over the non-empty subhypergraphs of Γ. An inductive branch decomposition has to satisfy additional conditions that ensure that "glueing" Γ_1 and Γ_2 together yields Γ.

Definition 3.14. Let $\Gamma = ((V, E), X)$ be a hypergraph with sources. An *inductive branch decomposition* of Γ is $T \in T_\Gamma$ where either:
- Γ is discrete (i.e. it has no edges) and $T = ()$;
- Γ has one edge and $T = (()—\Gamma—())$. We will use the shorthand $T = (\Gamma)$ in this case;
- $T = (T_1—\Gamma—T_2)$ and $T_i \in T_{\Gamma_i}$ are inductive branch decompositions of subhypergraphs $\Gamma_i = ((V_i, E_i), X_i)$ of Γ such that:
  - The edges are partitioned in two, $E = E_1 \sqcup E_2$ and $V = V_1 \cup V_2$;
  - The sources are those vertices shared with the original sources as well as those shared with the other subhypergraph, $X_i = (V_1 \cap V_2) \cup (X \cap V_i)$.

Note that $\mathsf{ends}(E_i) \subseteq V_i$ and that not all subtrees of a decomposition T are themselves decompositions: only those T' that contain all the nodes in T that are below the root of T'. We call these *full* subtrees and indicate with $\lambda(T')$ the subhypergraph of Γ that T' is a decomposition of. We sometimes write $\Gamma_i = \lambda(T_i)$, $V_i = \mathsf{vertices}(\Gamma_i)$ and $X_i = \mathsf{sources}(\Gamma_i)$. Then,

$$\mathsf{sources}(\Gamma_i) = (\mathsf{vertices}(\Gamma_1) \cap \mathsf{vertices}(\Gamma_2)) \cup (\mathsf{sources}(\Gamma) \cap \mathsf{vertices}(\Gamma_i)). \qquad (3.1)$$

Definition 3.15. Let $T = (T_1—\Gamma—T_2)$ be an inductive branch decomposition of $\Gamma = (G, X)$, with T_i possibly both empty. Define the *width* of T inductively: $\mathsf{wd}(()) := 0$, and $\mathsf{wd}(T) := \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), |\mathsf{sources}(\Gamma)|\}$. Expanding this expression, we obtain

$$\mathsf{wd}(T) = \max_{T' \text{ full subtree of } T} |\mathsf{sources}(\lambda(T'))|.$$

The *inductive branch width* of Γ is defined by the min-max formula $\mathsf{ibwd}(\Gamma) := \min_T \mathsf{wd}(T)$.

We show that this definition is equivalent to the original one by exhibiting a mapping from branch decompositions to inductive branch decompositions that preserve the width

and vice versa. Showing that these mappings preserve the width is a bit involved because the order of the edges in a decomposition is defined "globally", while, for an inductive decomposition, the width is defined inductively. Thus, we first need to show that we can compute the inductive width globally.

Lemma 3.16. *Let $\Gamma = (G, X)$ be a hypergraph with sources and T be an inductive branch decomposition of Γ. Let T_0 be a full subtree of T and let $T' \not\geq T_0$ denote a full subtree T' of T such that its intersection with T_0 is empty. Then,*

$$\mathsf{sources}(\lambda(T_0)) = \mathsf{vertices}(\lambda(T_0)) \cap \left(X \cup \bigcup_{T' \not\geq T_0} \mathsf{vertices}(\lambda(T')) \right).$$

Proof. Proceed by induction on the decomposition tree T. If it is empty, $T = ()$, then its subtree is also empty, $T_0 = ()$, and we are done.

If $T = (T_1 — \Gamma — T_2)$, then either T_0 is a full subtree of T_1, or it is a full subtree of T_2, or it coincides with T. If T_0 coincides with T, then their boundaries coincide and the statement is satisfied because $\mathsf{sources}(\lambda(T_0)) = X = V \cap X$. Now suppose that T_0 is a full subtree of T_1. Then, by applying the induction hypothesis, Equation (3.1), and using the fact that $\lambda(T_0) \subseteq \lambda(T_1)$, we compute the sources of T_0:

$\mathsf{sources}(\lambda(T_0))$

$= \mathsf{vertices}(\lambda(T_0)) \cap \left(\mathsf{sources}(\lambda(T_1)) \cup \bigcup_{T' \leq T_1, T' \not\geq T_0} \mathsf{vertices}(\lambda(T')) \right)$

$= \mathsf{vertices}(\lambda(T_0)) \cap \left((\mathsf{vertices}(\lambda(T_1)) \cap (\mathsf{vertices}(\lambda(T_2)) \cup X)) \cup \bigcup_{T' \leq T_1, T' \not\geq T_0} \mathsf{vertices}(\lambda(T')) \right)$

$= \mathsf{vertices}(\lambda(T_0)) \cap \left(\mathsf{vertices}(\lambda(T_2)) \cup X \cup \bigcup_{T' \leq T_1, T' \not\geq T_0} \mathsf{vertices}(\lambda(T')) \right)$

$= \mathsf{vertices}(\lambda(T_0)) \cap \left(X \cup \bigcup_{T' \leq T, T' \not\geq T_0} \mathsf{vertices}(\lambda(T')) \right)$

A similar computation can be done if T_0 is a full subtree of T_2. $\square$

Lemma 3.17. *Let $\Gamma = (G, X)$ be a hypergraph with sources and $G = (V, E)$ be its underlying hypergraph. Let T be an inductive branch decomposition of Γ. Then, there is a branch decomposition $\mathcal{I}^\dagger(T)$ of G such that $\mathsf{wd}(\mathcal{I}^\dagger(T)) \leq \mathsf{wd}(T)$.*

Proof. A binary tree is, in particular, a subcubic tree. Then, we can define Y to be the unlabelled tree underlying T. The label of a leaf l of T is a subhypergraph of Γ with one edge e_l. Then, there is a bijection $b \colon \mathsf{leaves}(T) \to \mathsf{edges}(G)$ such that $b(l) := e_l$. Then, (Y, b) is a branch decomposition of G and we can define $\mathcal{I}^\dagger(T) := (Y, b)$.

By construction, $e \in \mathsf{edges}(Y)$ if and only if $e \in \mathsf{edges}(T)$. Let $\{v, w\} = \mathsf{ends}(e)$ with v parent of w in T and let T_w the full subtree of T with root w. Let $\{E_v, E_w\}$ be the (non-trivial) partition of E induced by e. Then, for the edges sets, $E_w = \mathsf{edges}(\lambda(T_w))$ and $E_v = \bigcup_{T' \not\geq T_w} \mathsf{edges}(\lambda(T'))$, and, for the vertices sets, $\mathsf{ends}(E_w) \subseteq \mathsf{vertices}(\lambda(T_w))$ and

$\mathsf{ends}(E_v) \subseteq \bigcup_{T' \not\geq T_w} \mathsf{vertices}(\lambda(T'))$. Using these inclusions and applying Lemma 3.16,

$$
\begin{aligned}
\mathsf{ord}(e) && \mathsf{wd}(Y,b) \\
:= |\mathsf{ends}(E_w) \cap \mathsf{ends}(E_v)| && := \max_{e \in \mathsf{edges}(Y)} \mathsf{ord}(e) \\
\leq |\mathsf{vertices}(\lambda(T_w)) \cap \bigcup_{T' \not\geq T_w} \mathsf{vertices}(\lambda(T'))| && \leq \max_{T' < T} |\mathsf{sources}(\lambda(T'))| \\
\leq |\mathsf{vertices}(\lambda(T_w)) \cap (X \cup \bigcup_{T' \not\geq T_w} \mathsf{vertices}(\lambda(T')))| && \leq \max_{T' \leq T} |\mathsf{sources}(\lambda(T'))| \\
= |\mathsf{sources}(\lambda(T_w))| && = \mathsf{wd}(T) \qquad \square
\end{aligned}
$$

Lemma 3.18. *Let $\Gamma = (G, X)$ be a hypergraph with sources and $G = (V, E)$ be its underlying hypergraph. Let (Y, b) be a branch decomposition of G. Then, there is a branch decomposition $\mathcal{I}(Y, b)$ of Γ such that $\mathsf{wd}(\mathcal{I}(Y, b)) \leq \mathsf{wd}(Y, b) + |X|$.*

Proof. Proceed by induction on $|\mathsf{edges}(Y)|$. If Y has no edges, then either G has no edges and $(Y, b) = ()$ or G has only one edge e_l and $(Y, b) = (e_l)$. In either case, define $\mathcal{I}(Y, b) := (\Gamma)$ and $\mathsf{wd}(\mathcal{I}(Y, b)) := |X| \leq \mathsf{wd}(Y, b) + |X|$.

If Y has at least one edge e, then $Y = Y_1 \overset{e}{\text{—}} Y_2$ with Y_i a subcubic tree. Let $E_i = b(\mathsf{leaves}(Y_i))$ be the sets of edges of G indicated by the leaves of Y_i. Then, $E_1 \sqcup E_2 = E$. By induction hypothesis, there are inductive branch decompositions $T_i := \mathcal{I}(Y_i, b_i)$ of $\Gamma_i = (G_i, X_i)$, where $V_1 := \mathsf{ends}(E_1)$, $V_2 := \mathsf{ends}(E_2) \cup (V \setminus V_1)$, $X_i := (V_1 \cap V_2) \cup (V_i \cap X)$ and $G_i := (V_i, E_i)$. Then, the tree $\mathcal{I}(Y, b) := (T_1 \text{—} \Gamma \text{—} T_2)$ is an inductive branch decomposition of Γ and, by applying Lemma 3.16,

$$
\begin{aligned}
\mathsf{wd}(\mathcal{I}(Y,b)) \\
:= \max\{\mathsf{wd}(T_1), |X|, \mathsf{wd}(T_2)\} \\
= \max_{T' \leq T} |\mathsf{sources}(\lambda(T'))| \\
\leq \max_{T' \leq T} |\mathsf{vertices}(\lambda(T')) \cap \mathsf{ends}(E \setminus \mathsf{edges}(\lambda(T')))| + |X| \\
= \max_{e \in \mathsf{edges}(Y)} \mathsf{ord}(e) + |X| \\
=: \mathsf{wd}(Y,b) + |X| \qquad \square
\end{aligned}
$$

Combining Lemma 3.17 and Lemma 3.18 we obtain:

Proposition 3.19. *Inductive branch width is equivalent to branch width.*

3.3. **Cospans of hypergraphs.** We work with the category $\mathsf{UHGraph}$ of undirected hypergraphs and their homomorphisms (Definition 3.1). The monoidal category $\mathsf{Cospan}(\mathsf{UHGraph})$ of cospans is a standard choice for an algebra of "open" hypergraphs. Hypergraphs are composed by glueing vertices [RSW05, GH97, Fon15]. We do not need the full expressivity of $\mathsf{Cospan}(\mathsf{UHGraph})$ and restrict to $\mathsf{Cospan}(\mathsf{UHGraph})_*$, where the objects are sets, seen as discrete hypergraphs.

Definition 3.20. A *cospan* in a category C is a pair of morphisms in C that share the same codomain, called the *head*, $f \colon X \to E$ and $g \colon Y \to E$. When C has finite colimits, cospans form a symmetric monoidal category $\mathsf{Cospan}(\mathsf{C})$ whose objects are the objects of C

and morphisms are cospans in C. More precisely, a morphism $X \to Y$ in $\mathsf{Cospan}(\mathsf{C})$ is an equivalence class of cospans $X \xrightarrow{f} E \xleftarrow{g} Y$, up to isomorphism of the head of the cospan. The composition of $X \xrightarrow{f} E \xleftarrow{g} Y$ and $Y \xrightarrow{h} F \xleftarrow{l} Z$ is given by the pushout of g and h. Intuitively, the pushout of g and h "glues" E and F along the images of g and h (see Example 3.23). The monoidal product is given by component-wise coproducts.

We can construct the category of cospans of hypergraphs $\mathsf{Cospan}(\mathsf{UHGraph})$ because the category of hypergraphs $\mathsf{UHGraph}$ has all finite colimits.

Proposition 3.21. *The category* $\mathsf{UHGraph}$ *has all finite colimits and they are computed pointwise.*

Proof. Let $\mathbf{D}\colon \mathsf{J} \to \mathsf{UHGraph}$ be a diagram in $\mathsf{UHGraph}$. Then, every object i in J determines a hypergraph $G_i := \mathbf{D}(i) = (V_i, E_i, \mathsf{ends}_i)$ and every $f\colon i \to j$ in J, gives a hypergraph homomorphism $\mathbf{D}(f) = (f_V, f_E)$. Let the functors $\mathbf{U}_E\colon \mathsf{UHGraph} \to \mathsf{Set}$ and $\mathbf{U}_V\colon \mathsf{UHGraph} \to \mathsf{Set}$ associate the edges, resp. vertices, component to hypergraphs and hypergraph homomorphisms: for a hypergraph $G = (V, E)$, $\mathbf{U}_E(G) := E$ and $\mathbf{U}_V(G) := V$; and, for a morphism $f = (f_V, f_E)$, $\mathbf{U}_E(f) := f_E$ and $\mathbf{U}_V(f) := f_V$.

$$
\left(i \xrightarrow{f} j\right) \xrightarrow{\;\mathbf{D}\;} \left(G_i \xrightarrow{(f_V, f_E)} G_j\right) \nearrow^{\mathbf{U}_V} \left(V_i \xrightarrow{f_V} V_j\right) \searrow_{\mathbf{U}_E} \left(E_i \xrightarrow{f_E} E_j\right)
$$

The category Set has all colimits, thus there are $E_0 := \mathrm{colim}(\mathbf{D}; \mathbf{U}_E)$ and $V_0 := \mathrm{colim}(\mathbf{D}; \mathbf{U}_V)$. Let $c_i\colon V_i \to V_0$ and $d_i\colon E_i \to E_0$ be the inclusions given by the colimits. Then, for any $i, j \in \mathsf{Obj}(\mathsf{J})$ the following diagrams commute:
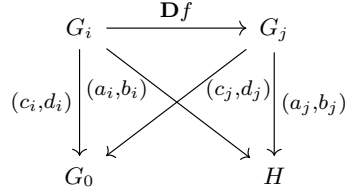
$$
\begin{array}{ccc}
V_i \xrightarrow{f_V} V_j & \qquad & E_i \xrightarrow{f_E} E_j \\
{}_{c_i}\searrow \quad \swarrow_{c_j} & & {}_{d_i}\searrow \quad \swarrow_{d_j} \\
V_0 & & E_0
\end{array}
$$

By definition of hypergraph morphism, $f_E \,;\, \mathsf{ends}_j = \mathsf{ends}_i \,;\, \wp(f_V)$, and, by functoriality of $\wp$, $\wp(f_V) \,;\, \wp(c_j) = \wp(c_i)$. This shows that $\wp(V_0)$ is a cocone over $\mathbf{D} \,;\, \mathbf{U}_E$ with morphisms given by $\mathsf{ends}_i \,;\, \wp(c_i)$. Then, there is a unique morphism $\mathsf{ends}\colon E_0 \to \wp(V_0)$ that commutes with the cocone morphisms: $d_i \,;\, \mathsf{ends} = \mathsf{ends}_i \,;\, \wp(c_i)$.

$$
\begin{array}{ccccc}
& \overset{\mathsf{ends}_i}{\frown} & & \overset{\mathsf{ends}_j}{\frown} & \\
E_i \xrightarrow{f_E} E_j & & \wp(V_i) \xrightarrow{\wp(f_V)} \wp(V_j) \\
{}_{d_i}\searrow \quad \swarrow_{d_j} & & {}_{\wp(c_i)}\searrow \quad \swarrow_{\wp(c_j)} \\
E_0 \dashrightarrow{\mathsf{ends}} & & \wp(V_0)
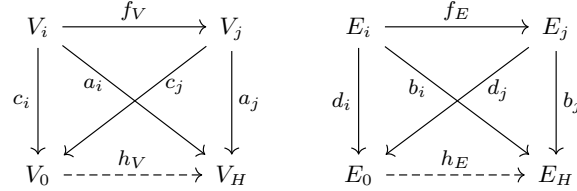\end{array}
$$

This shows that the pairs (c_i, d_i) are hypergraph morphisms and, with the hypergraph defined by $G_0 := (V_0, E_0, \mathsf{ends})$, form a cocone over $\mathbf{D}$ in $\mathsf{UHGraph}$. Let $H = (V_H, E_H, \mathsf{ends}_H)$ be another cocone over $\mathbf{D}$ with morphisms $(a_i, b_i) \colon G_i \to H$.
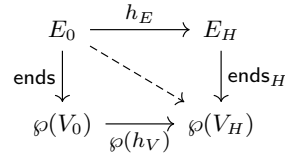

We show that G_0 is initial by constructing a morphism $(h_V, h_E) \colon G_0 \to H$ and showing that it is the unique one commuting with the inclusions.

By applying the functors $\mathbf{U}_E$ and $\mathbf{U}_V$ to the diagram above, we obtain the following diagrams in Set, where $h_V \colon V_0 \to V_H$ and $h_E \colon E_0 \to E_H$ are the unique morphism from the colimit cone.

We show that (h_V, h_E) is a hypergraph morphism. The object $\wp(V_H)$ is a cocone over $\mathbf{D} \mathbin{;} \mathbf{U}_E$ in (at least) two ways: with morphisms $d_i \mathbin{;} \mathsf{ends} \mathbin{;} \wp(h_V)$ and morphisms $b_i \mathbin{;} \mathsf{ends}_H$. By initiality of E_0, there is a unique morphism $E_0 \to \wp(V_H)$ and it must coincide with $h_E \mathbin{;} \mathsf{ends}_H$ and $\mathsf{ends} \mathbin{;} \wp(h_V)$.

This proves that (h_V, h_E) is a hypergraph morphism. It is, moreover, unique because any other morphism with this property would have the same components. In fact, let $(h'_V, h'_E) \colon G_0 \to H$ be another hypergraph morphism that commutes with the cocones, i.e. $(c_i, d_i) \mathbin{;} (h'_V, h'_E) = (a_i, b_i)$. Then, its components must commute with the respective cocones in Set, by functoriality of $\mathbf{U}_E$ and $\mathbf{U}_V$: $c_i \mathbin{;} h'_V = a_i$ and $d_i \mathbin{;} h'_E = b_i$. By construction, V_0 and E_0 are the colimits of $\mathbf{D} \mathbin{;} \mathbf{U}_V$ and $\mathbf{D} \mathbin{;} \mathbf{U}_E$, so there are unique morphisms to any other cocone over the same diagrams. This means that $h'_V = h_V$ and $h'_E = h_E$, which shows the uniqueness of (h_V, h_E). $\qquad\square$

Definition 3.22. The category $\mathsf{Cospan}(\mathsf{UHGraph})_*$ is the full subcategory of $\mathsf{Cospan}(\mathsf{UHGraph})$ on discrete hypergraphs. Objects are sets and a morphism $g \colon X \to Y$ is given by a hypergraph $G = (V, E)$ and two functions, $\partial_X \colon X \to V$ and $\partial_Y \colon Y \to V$.

Composition in $\mathsf{Cospan}(\mathsf{UHGraph})_*$ is given by identification of the common sources: if two vertices are pointed by a common source, then they are identified.

Example 3.23. The composition of two morphisms with a single edge along a common vertex gives a path of length two, obtained by identifying the vertex v of the first morphism
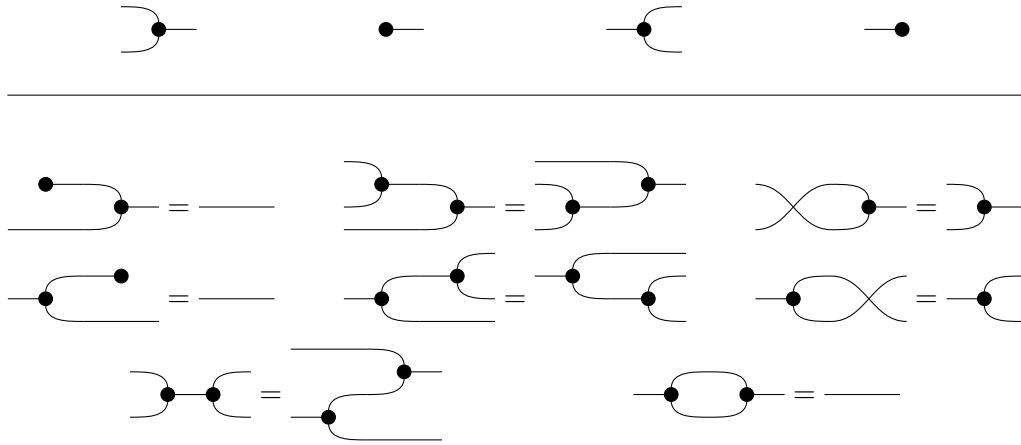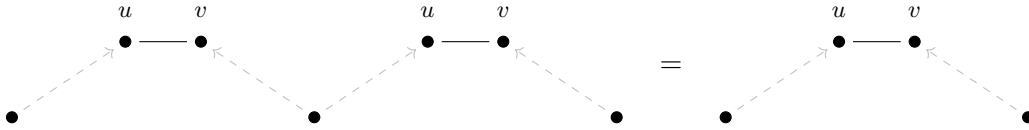
FIGURE 5. Generators and axioms of a special Frobenius monoid.

with the vertex u of the second.

3.4. **String diagrams for cospans of hypergraphs.** We introduce a syntax for the monoidal category $\mathsf{Cospan}(\mathsf{UHGraph})_*$, which we will use for proving some of the results in this section. We will show that the syntax for $\mathsf{Cospan}(\mathsf{UHGraph})_*$ is given by the syntax of $\mathsf{Cospan}(\mathsf{Set})$ together with an extra "hyperedge" generator $\mathsf{e}_n \colon n \to 0$ for every $n \in \mathbb{N}$. This result is inspired by the similar one for cospans of directed graphs [RSW05].

It is well-known that the category $\mathsf{Cospan}(\mathsf{Set})$ of finite sets and cospans of functions between them has a convenient syntax given by the walking special Frobenius monoid [Lac04].

Proposition 3.24 [Lac04]**.** *The skeleton of the monoidal category* $\mathsf{Cospan}(\mathsf{Set})$ *is isomorphic to the prop* sFrob, *whose generators and axioms are in Figure 5.*

In order to obtain cospans of hypergraphs from cospans of sets, we need to add generators that behave like hyperedges: they have n inputs and these inputs can be permuted without any effect.

Definition 3.25. Define UHedge to be the prop generated by a "hyperedge" generator $\mathsf{e}_n \colon n \to 0$ for every $n \in \mathbb{N}$ such that permuting its inputs does not have any effect:

$$\forall n \in \mathbb{N} \quad \overset{n}{-\!\bigcirc} \quad \text{such that} \quad \forall \text{ permutation } \sigma \colon n \to n \quad n-\boxed{\sigma}\overset{n}{\vdash\!\bigcirc} = \overset{n}{-\!\bigcirc}$$

The syntax for cospans of graphs is defined as a coproduct of props.

Definition 3.26. Define the prop FGraph as a coproduct: $\mathsf{FGraph} \coloneqq \mathsf{sFrob} + \mathsf{UHedge}$.

We will show that every morphism $g \colon n \to m$ in FGraph corresponds to a morphism in $\mathsf{Cospan}(\mathsf{UHGraph})_*$.

Example 3.27. The string diagram below corresponds to a hypergraph with two left sources, one right source and two hyperedges. The number of endpoints of each hyperedge is given by the arity of the corresponding generator in the string diagram. Two hyperedges are adjacent to the same vertex when they are connected by the Frobenius structure in the string diagram, and a hyperedge is adjacent to a source when it is connected to an input or output in the string diagram.

Proposition 3.28. *There is a symmetric monoidal functor* $\mathbf{S}\colon \mathsf{FGraph} \to \mathsf{Cospan}(\mathsf{UHGraph})_*$.

Proof. By definition, $\mathsf{FGraph} \coloneqq \mathsf{sFrob} + \mathsf{UHedge}$ is a coproduct. Therefore, it suffices to define two symmetric monoidal functors $\mathbf{S_1}\colon \mathsf{sFrob} \to \mathsf{Cospan}(\mathsf{UHGraph})_*$ and $\mathbf{S_2}\colon \mathsf{UHedge} \to \mathsf{Cospan}(\mathsf{UHGraph})_*$ for constructing the functor $\mathbf{S} \coloneqq [\mathbf{S_1}, \mathbf{S_2}]$.

The category of cospans of finite sets embeds into the category of cospans of undirected hypergraphs, and in particular $\mathsf{Cospan}(\mathsf{Set}) \hookrightarrow \mathsf{Cospan}(\mathsf{UHGraph})_*$. By Proposition 3.24, there is a functor $\mathsf{sFrob} \to \mathsf{Cospan}(\mathsf{Set})$, which gives us a functor $\mathbf{S_1}\colon \mathsf{sFrob} \to \mathsf{Cospan}(\mathsf{UHGraph})_*$.

For the functor $\mathbf{S_2}$, we need to define it on the generators of UHedge and show that it preserves the equations. We define $\mathbf{S_2}(\mathsf{e}_n)$ to be the cospan of graphs $n \to (n, \{e\}) \leftarrow \emptyset$ given by $\mathbb{1}_n\colon n \to n$ and $\mathsf{i}_n\colon \emptyset \to n$. With this assignment, we can freely extend $\mathbf{S_2}$ to a monoidal functor $\mathsf{UHedge} \to \mathsf{Cospan}(\mathsf{UHGraph})_*$. In fact, it preserves the equations of UHedge because permuting the order of the endpoints of an undirected hyperedge has no effect by definition. $\qquad\square$
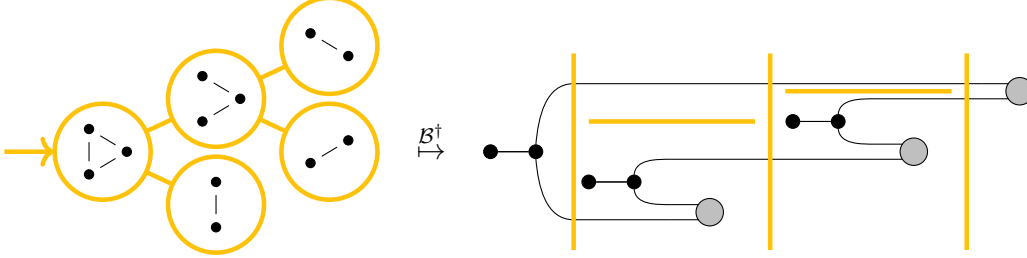
In order to instantiate monoidal width in $\mathsf{Cospan}(\mathsf{UHGraph})_*$, we need to define an appropriate weight function.

Definition 3.29. Let $\mathcal{A}$ be all morphisms of $\mathsf{Cospan}(\mathsf{UHGraph})_*$. Define the *weight function* as follows. For an object X, $\mathsf{w}(X) \coloneqq |X|$. For a morphism $g \in \mathcal{A}$, $\mathsf{w}(g) \coloneqq |V|$, where V is the set of vertices of the apex of g, i.e. $g = X \to G \leftarrow Y$ and $G = (V, E)$.

3.5. **Tree width as monoidal width.** Here we show that monoidal width in the monoidal category $\mathsf{Cospan}(\mathsf{UHGraph})_*$, with the weight function given in Definition 3.29, is equivalent to tree width. We do this by bounding monoidal width by above with branch width $+1$ and by below with half of branch width (Theorem 3.34). We prove these bounds by defining maps from inductive branch decompositions to monoidal decompositions that preserve the width (Proposition 3.30), and vice versa (Proposition 3.33).

The idea behind the mapping from inductive branch decompositions to monoidal decompositions is to take a one-edge hypergraph for each leaf of the inductive branch decomposition and compose them following the structure of the decomposition tree. The 3-clique has a branch decomposition as shown on the left. The corresponding monoidal
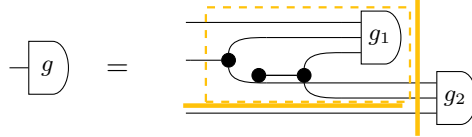
decomposition is shown on the right.

Proposition 3.30. *Let $\Gamma = (G, X)$ be a hypergraph with sources and T be an inductive branch decomposition of Γ. Let $g \coloneqq X \xrightarrow{\iota} G \leftarrow \emptyset$ be the corresponding cospan. Then, there is a monoidal decomposition $\mathcal{B}^\dagger(T) \in D_g$ such that $\mathsf{wd}(\mathcal{B}^\dagger(T)) \leq \max\{\mathsf{wd}(T) + 1, \gamma(G)\}$.*

Proof. Let $G = (V, E)$ and proceed by induction on the decomposition tree T. If the tree $T = (\Gamma)$ is composed of only a leaf, then the label Γ of this leaf must have only one hyperedge with $\gamma(G)$ endpoints and $\mathsf{wd}(T) \coloneqq |X|$. We define the corresponding monoidal decomposition to also consist of only a leaf, $\mathcal{B}^\dagger(T) \coloneqq (g)$, and obtain the desired bound $\mathsf{wd}(\mathcal{B}^\dagger(T)) = \max\{|X|, \gamma(G)\} = \max\{\mathsf{wd}(T), \gamma(G)\}$.

If $T = (T_1{-}\Gamma{-}T_2)$, then, by definition of branch decomposition, T is composed of two subtrees T_1 and T_2 that give branch decompositions of $\Gamma_1 = (G_1, X_1)$ and $\Gamma_2 = (G_2, X_2)$. There are three conditions imposed by the definition on these subgraphs $G_i = (V_i, E_i)$: $E = E_1 \sqcup E_2$ with $E_i \neq \emptyset$, $V_1 \cup V_2 = V$, and $X_i = (V_1 \cap V_2) \cup (X \cap V_i)$. Let $g_i = X_i \to G_i \leftarrow \emptyset$ be the cospan given by $\iota \colon X_i \to V_i$ and corresponding to Γ_i. Then, we can decompose g in terms of identities, the structure of $\mathsf{Cospan(UHGraph)}_*$, and its subgraphs g_1 and g_2:

By induction hypothesis, there are monoidal decompositions $\mathcal{B}^\dagger(T_i)$ of g_i whose width is bounded: $\mathsf{wd}(\mathcal{B}^\dagger(T_i)) \leq \max\{\mathsf{wd}(T_i) + 1, \gamma(G_i)\}$. By Lemma 2.8, there is a monoidal decomposition $\mathcal{C}(\mathcal{B}^\dagger(T_1))$ of the morphism in the above dashed box of bounded width: $\mathsf{wd}(\mathcal{C}(\mathcal{B}^\dagger(T_1))) \leq \max\{\mathsf{wd}(\mathcal{B}^\dagger(T_1)), |X_1| + 1\}$. Using this decomposition, we can define the monoidal decomposition given by the cuts in the figure above.

$$\mathcal{B}^\dagger(T) \coloneqq ((\mathcal{C}(\mathcal{B}^\dagger(T_1)){-} \otimes {-}\mathbb{1}_{X_2 \setminus X_1}){-};_{X_2}{-}\mathcal{B}^\dagger(T_2)).$$

We can bound its width by applying Lemma 2.8, the induction hypothesis and the relevant definitions of width (Definition 3.11 and Definition 3.29).

$$
\begin{aligned}
&\mathsf{wd}(\mathcal{B}^\dagger(T)) \\
&\coloneqq \max\{\mathsf{wd}(\mathcal{C}(\mathcal{B}^\dagger(T_1))), \mathsf{wd}(\mathcal{B}^\dagger(T_2)), |X_2|\} \\
&= \max\{\mathsf{wd}(\mathcal{B}^\dagger(T_1)), \mathsf{wd}(\mathcal{B}^\dagger(T_2)), |X_1| + 1, |X_2|\} \\
&\leq \max\{\mathsf{wd}(T_1) + 1, \gamma(G_1), \mathsf{wd}(T_2) + 1, \gamma(G_2), |X_1| + 1, |X_2|\} \\
&\leq \max\{\max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), |X_1|, |X_2|\} + 1, \gamma(G_1), \gamma(G_2)\} \\
&\leq \max\{\max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), |X|\} + 1, \gamma(G)\} \\
&\eqqcolon \max\{\mathsf{wd}(T) + 1, \gamma(G)\} \qquad\qquad\qquad\qquad\qquad\qquad \square
\end{aligned}
$$

The mapping $\mathcal{B}$ follows the same idea of the mapping $\mathcal{B}^\dagger$ but requires extra care: we need to keep track of which vertices are going to be identified in the final cospan. The function ϕ stores this information, thus it cannot identify two vertices that are not already in the boundary of the hypergraph. The proof of Proposition 3.33 proceeds by induction on the monoidal decomposition and constructs the corresponding branch decomposition. The inductive step relies on ϕ to identify which subgraphs of Γ correspond to the two subtrees in the monoidal decomposition, and, consequently, to define the corresponding branch decomposition.

Remark 3.31. Let $f\colon A \to C$ and $g\colon B \to C$ be two functions. The union of the images of f and g is the image of the coproduct map $[f,g]\colon A+B \to C$, i.e. $\mathsf{im}(f)\cup\mathsf{im}(g) = \mathsf{im}([f,g])$. The intersection of the images of f and g is the image of the pullback map $\langle f \wedge g \rangle\colon A \times_C B \to C$, i.e. $\mathsf{im}(f) \cap \mathsf{im}(g) = \mathsf{im}(\langle f \wedge g \rangle)$.

Remark 3.32. Let $f\colon A \to C$, $g\colon B \to C$ and $\phi\colon C \to V$ such that $\forall\ c \neq c' \in C\ \phi(c) = \phi(c') \Rightarrow c, c' \in \mathsf{im}(f)$. We have that $\mathsf{im}(\langle f\,;\phi\wedge g\,;\phi\rangle) \supseteq \mathsf{im}(\langle f\wedge g\rangle\,;\phi)$. Then, $\mathsf{im}(\langle f\,;\phi\wedge g\,;\phi\rangle) = \mathsf{im}(\langle f \wedge g \rangle\,;\phi)$ because their difference is empty:

$$\mathsf{im}(\langle f \wedge g \rangle\,;\phi) \setminus \mathsf{im}(\langle f\,;\phi \wedge g\,;\phi\rangle)$$
$$= \{v \in V : \exists a \in A\ \exists b \in B\ \phi(f(a)) = \phi(g(b)) \wedge f(a) \notin \mathsf{im}(g) \wedge g(b) \notin \mathsf{im}(f)\} = \emptyset$$
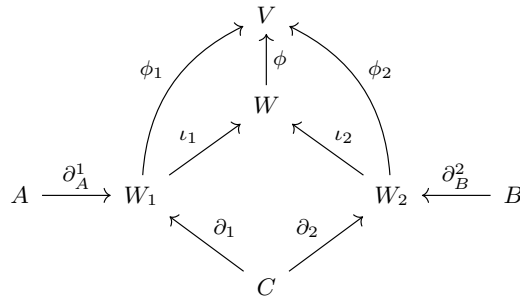
Proposition 3.33. *Let* $h = A \xrightarrow{\partial_A} H \xleftarrow{\partial_B} B$ *with* $H = (W, F)$. *Let* $\phi\colon W \to V$ *such that* $\forall\ w \neq w' \in W\ \phi(w) = \phi(w') \Rightarrow w, w' \in \mathsf{im}(\partial_A) \cup \mathsf{im}(\partial_B)$ *(glueing property). Let* d *be a monoidal decomposition of* h. *Let* $\Gamma := ((\mathsf{im}(\phi), F), \mathsf{im}(\partial_A\,;\phi) \cup \mathsf{im}(\partial_B\,;\phi))$. *Then, there is an inductive branch decomposition* $\mathcal{B}(d)$ *of* Γ *such that* $\mathsf{wd}(\mathcal{B}(d)) \leq 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$.

Proof. Proceed by induction on the decomposition tree d. If it is just a leaf, $d = (h)$ and H has no edges, $F = \emptyset$, then the corresponding inductive branch decomposition is empty, $\mathcal{B}(d) := ()$, and we can compute its width: $\mathsf{wd}(\mathcal{B}(d)) := 0 \leq 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$.

If the decomposition is just a leaf $d = (h)$ but H has exactly one edge, $F = \{e\}$, then the corresponding branch decomposition is just a leaf as well, $\mathcal{B}(d) := (\Gamma)$, and we can compute its width: $\mathsf{wd}(\mathcal{B}(d)) := |\mathsf{im}(\partial_A\,;\phi) \cup \mathsf{im}(\partial_B\,;\phi)| \leq |A| + |B| \leq 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$.

If the decomposition is just a leaf $d = (h)$ and H has more than one edge, $|F| > 1$, then we can let $\mathcal{B}(d)$ be any inductive branch decomposition of Γ. Its width is not greater than the number of vertices in Γ, thus we can bound its width $\mathsf{wd}(\mathcal{B}(d)) \leq |\mathsf{im}(\phi)| \leq 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$.

If $d = (d_1 \text{---}\,;_C\,\text{---} d_2)$, then d_i is a monoidal decomposition of h_i with $h = h_1\,;_C h_2$. We can give the expressions of these morphisms: $h_1 = A \xrightarrow{\partial_A^1} H_1 \xleftarrow{\partial_1} C$ and $h_2 = C \xrightarrow{\partial_2} H_2 \xleftarrow{\partial_B^2} B$, with $H_i = (W_i, F_i)$, and obtain the following diagram, where $\iota_i\colon W_i \to W$ are the functions induced by the pushout and we define $\phi_i := \iota_i\,;\phi$.

We show that ϕ_1 satisfies the glueing property in order to apply the induction hypothesis to ϕ_1 and H_1: let $w \neq w' \in W_1$ such that $\phi_1(w) = \phi_1(w')$. Then, $\iota_1(w) = \iota_1(w')$ or $\phi(\iota_1(w)) = \phi(\iota_1(w')) \wedge \iota_1(w) \neq \iota_1(w')$. Then, $w, w' \in \mathsf{im}(\partial_1)$ or $\iota_1(w), \iota_1(w') \in \mathsf{im}(\partial_A \, ; \phi) \cup \mathsf{im}(\partial_B \, ; \phi)$. Then, $w, w' \in \mathsf{im}(\partial_1)$ or $w, w' \in \mathsf{im}(\partial_A^1)$. Then, $w, w' \in \mathsf{im}(\partial_1) \cup \mathsf{im}(\partial_A^1)$. Similarly, we can show that ϕ_2 satisfies the same property. Then, we can apply the induction hypothesis to get an inductive branch decomposition $\mathcal{B}(d_1)$ of $\Gamma_1 = ((\mathsf{im}(\phi_1), F_1), \mathsf{im}(\partial_A^1 \, ; \phi_1) \cup \mathsf{im}(\partial_1 \, ; \phi_1))$ and an inductive branch decomposition $\mathcal{B}(d_2)$ of $\Gamma_2 = ((\mathsf{im}(\phi_2), F_2), \mathsf{im}(\partial_B^2 \, ; \phi_2) \cup \mathsf{im}(\partial_2 \, ; \phi_2))$ with bounded width: $\mathsf{wd}(\mathcal{B}(d_1)) \leq 2 \cdot \max\{\mathsf{wd}(d_1), |A|, |C|\}$ and $\mathsf{wd}(\mathcal{B}(d_2)) \leq 2 \cdot \max\{\mathsf{wd}(d_2), |B|, |C|\}$.

We check that we can define an inductive branch decomposition of Γ from $\mathcal{B}(d_1)$ and $\mathcal{B}(d_2)$.

- $F = F_1 \sqcup F_2$ because the pushout is along discrete hypergraphs.
- $\mathsf{im}(\phi) = \mathsf{im}(\phi_1) \cup \mathsf{im}(\phi_2)$ because $\mathsf{im}([\iota_1, \iota_2]) = W$ and $\mathsf{im}(\phi_1) \cup \mathsf{im}(\phi_2) = \mathsf{im}(\iota_1 \, ; \phi) \cup \mathsf{im}(\iota_2 \, ; \phi) = \mathsf{im}([\iota_1, \iota_2] \, ; \phi) = \mathsf{im}(\phi)$.
- $\mathsf{im}([\partial_A^1, \partial_1] \, ; \phi_1) = \mathsf{im}(\phi_1) \cap (\mathsf{im}(\phi_2) \cup \mathsf{im}(\partial_A \, ; \phi) \cup \mathsf{im}(\partial_B \, ; \phi))$ because

$$\mathsf{im}(\phi_1) \cap (\mathsf{im}(\phi_2) \cup \mathsf{im}(\partial_A \, ; \phi) \cup \mathsf{im}(\partial_B \, ; \phi))$$

$$= \quad \text{(by definition of } \phi_i)$$

$$\mathsf{im}(\iota_1 \, ; \phi) \cap (\mathsf{im}(\iota_2 \, ; \phi) \cup \mathsf{im}(\partial_A \, ; \phi) \cup \mathsf{im}(\partial_B \, ; \phi))$$

$$= \quad (\text{because } \mathsf{im}(\partial_B) = \mathsf{im}(\partial_B^2 \, ; \iota_2) \subseteq \mathsf{im}(\iota_2))$$

$$\mathsf{im}(\iota_1 \, ; \phi) \cap (\mathsf{im}(\iota_2 \, ; \phi) \cup \mathsf{im}(\partial_A \, ; \phi))$$

$$= \quad \text{(by Remark 3.31)}$$

$$\mathsf{im}(\iota_1 \, ; \phi) \cap \mathsf{im}([\iota_2, \partial_A] \, ; \phi)$$

$$= \quad \text{(by Remark 3.31)}$$

$$\mathsf{im}(\langle \iota_1 \, ; \phi \wedge [\iota_2, \partial_A] \, ; \phi \rangle)$$

$$= \quad \text{(by Remark 3.32)}$$

$$\mathsf{im}(\langle \iota_1 \wedge [\iota_2, \partial_A] \rangle \, ; \phi)$$

$$= \quad \text{(because pullbacks commute with coproducts)}$$

$$\mathsf{im}([\langle \iota_1 \wedge \iota_2 \rangle, \langle \iota_1 \wedge \partial_A \rangle] \, ; \phi)$$

$$= \quad (\text{because } \partial_A = \partial_A^1 \, ; \iota_1)$$

$$\mathsf{im}([\langle \iota_1 \wedge \iota_2 \rangle, \partial_A] \, ; \phi)$$

$$= \quad (\text{because } \partial_1 \, ; \iota_1 = \partial_2 \, ; \iota_2 \text{ is the pushout map of } \partial_1 \text{ and } \partial_2)$$

$$\mathsf{im}([\partial_1 \, ; \iota_1, \partial_A^1 \, ; \iota_1] \, ; \phi)$$

$$= \quad \text{(by property of the coproduct)}$$

$$\mathsf{im}([\partial_1, \partial_A^1] \, ; \phi_1)$$

- $\mathsf{im}([\partial_2, \partial_B^2] \, ; \phi_2) = \mathsf{im}(\phi_2) \cap (\mathsf{im}(\phi_1) \cup \mathsf{im}(\partial_A \, ; \phi) \cup \mathsf{im}(\partial_B \, ; \phi))$ similarly to the former point.

Then, $\mathcal{B}(d) := (\mathcal{B}(d_1)\text{---}\Gamma\text{---}\mathcal{B}(d_2))$ is an inductive branch decomposition of Γ and

$$\mathsf{wd}(\mathcal{B}(d))$$

$$:= \max\{\mathsf{wd}(\mathcal{B}(d_1)), |\mathsf{im}([\partial_A, \partial_B])|, \mathsf{wd}(\mathcal{B}(d_2))\}$$

$$\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot |A|, 2 \cdot |C|, |A| + |B|,$$

$$2 \cdot \mathsf{wd}(d_2), 2 \cdot |B|\}$$
$$\leq 2 \cdot \max\{\mathsf{wd}(d_1), |A|, |C|, \mathsf{wd}(d_2), |B|\}$$
$$=: 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$$

If $d = (d_1 — \otimes — d_2)$, then d_i is a monoidal decomposition of h_i with $h = h_1 \otimes h_2$. Let $h_i = X_i \overset{\partial_X^i}{\to} H_i \overset{\partial_Y^i}{\leftarrow} Y_i$ with $H_i = F_i \overset{s,t}{\rightrightarrows} W_i$. Let $\iota_i \colon W_i \to W$ be the inclusions induced by the monoidal product. Define $\phi_i := \iota_i \,;\, \phi$. We show that ϕ_1 satisfies the glueing property: Let $w \neq w' \in W_1$ such that $\phi_1(w) = \phi_1(w')$. Then, $\iota_1(w) = \iota_1(w')$ or $\phi(\iota_1(w)) = \phi(\iota_1(w')) \wedge \iota_1(w) \neq \iota_1(w')$. Then, $\iota_1(w), \iota_1(w') \in \mathsf{im}(\partial_A \,;\, \phi) \cup \mathsf{im}(\partial_B \,;\, \phi)$ because ι_i are injective. Then, $w, w' \in \mathsf{im}(\partial_A^1) \cup \mathsf{im}(\partial_B^1)$. Similarly, we can show that ϕ_2 satisfies the same property. Then, we can apply the induction hypothesis to get $\mathcal{B}(d_i)$ inductive branch decomposition of $\Gamma_i = ((\mathsf{im}(\phi_i), F_i), \mathsf{im}([\partial_A^i, \partial_B^i] \,;\, \phi_i))$ such that $\mathsf{wd}(\mathcal{B}(d_i)) \leq 2 \cdot \max\{\mathsf{wd}(d_i), |A_i|, |B_i|\}$.

We check that we can define an inductive branch decomposition of Γ from $\mathcal{B}(d_1)$ and $\mathcal{B}(d_2)$.

- $F = F_1 \sqcup F_2$ because the monoidal product is given by the coproduct in Set.
- $\mathsf{im}(\phi) = \mathsf{im}(\phi_1) \cup \mathsf{im}(\phi_2)$ because $\mathsf{im}([\iota_1, \iota_2]) = W$ and $\mathsf{im}(\phi_1) \cup \mathsf{im}(\phi_2) = \mathsf{im}(\iota_1 \,;\, \phi) \cup \mathsf{im}(\iota_2 \,;\, \phi) = \mathsf{im}([\iota_1, \iota_2] \,;\, \phi) = \mathsf{im}(\phi)$.
- $\mathsf{im}([\partial_A^1, \partial_B^1] \,;\, \phi_1) = \mathsf{im}(\phi_1) \cap (\mathsf{im}(\phi_2) \cup \mathsf{im}(\partial_A \,;\, \phi) \cup \mathsf{im}(\partial_B \,;\, \phi))$ because

$$\mathsf{im}(\phi_1) \cap (\mathsf{im}(\phi_2) \cup \mathsf{im}(\partial_A \,;\, \phi) \cup \mathsf{im}(\partial_B \,;\, \phi))$$
$$= \quad \text{(by definition of } \phi_i)$$
$$\mathsf{im}(\iota_1 \,;\, \phi) \cap (\mathsf{im}(\iota_2 \,;\, \phi) \cup \mathsf{im}(\partial_A \,;\, \phi) \cup \mathsf{im}(\partial_B \,;\, \phi))$$
$$= \quad \text{(by Remark 3.31 and property of the coproduct)}$$
$$\mathsf{im}(\iota_1 \,;\, \phi) \cap \mathsf{im}([\iota_2, [\partial_A, \partial_B]] \,;\, \phi)$$
$$= \quad \text{(by Remark 3.31)}$$
$$\mathsf{im}(\langle \iota_1 \,;\, \phi \wedge [\iota_2, [\partial_A, \partial_B]] \,;\, \phi \rangle)$$
$$= \quad \text{(by Remark 3.32)}$$
$$\mathsf{im}(\langle \iota_1 \wedge [\iota_2, [\partial_A, \partial_B]] \rangle \,;\, \phi)$$
$$= \quad \text{(because pullbacks commute with coproducts)}$$
$$\mathsf{im}([\langle \iota_1 \wedge \iota_2 \rangle, \langle \iota_1 \wedge [\partial_A, \partial_B] \rangle] \,;\, \phi)$$
$$= \quad \text{(because } \langle \iota_1 \wedge \iota_2 \rangle = \mathsf{i})$$
$$\mathsf{im}(\langle \iota_1 \wedge [\partial_A, \partial_B] \rangle \,;\, \phi)$$
$$= \quad \text{(because } \partial_A = \partial_A^1 + \partial_A^2 \text{ and } \partial_B = \partial_B^1 + \partial_B^2)$$
$$\mathsf{im}([\partial_A^1 \,;\, \iota_1, \partial_B^1 \,;\, \iota_1] \,;\, \phi)$$
$$= \quad \text{(by property of the coproduct)}$$
$$\mathsf{im}([\partial_A^1, \partial_B^1] \,;\, \phi_1)$$

- $\mathsf{im}([\partial_A^2, \partial_B^2] \,;\, \phi_2) = \mathsf{im}(\phi_2) \cap (\mathsf{im}(\phi_1) \cup \mathsf{im}(\partial_A \,;\, \phi) \cup \mathsf{im}(\partial_B \,;\, \phi))$ similarly to the former point.

Then, $\mathcal{B}(d) := (\mathcal{B}(d_1) — \Gamma — \mathcal{B}(d_2))$ is an inductive branch decomposition of Γ and

$$\mathsf{wd}(\mathcal{B}(d))$$
$$:= \max\{\mathsf{wd}(\mathcal{B}(d_1)), |\mathsf{im}([\partial_A, \partial_B])|, \mathsf{wd}(\mathcal{B}(d_2))\}$$

$$\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot |A_1|, 2 \cdot |B_1|, |A| + |B|,$$
$$2 \cdot \mathsf{wd}(d_2), 2 \cdot |A_2|, 2 \cdot |B_2|\}$$
$$\leq 2 \cdot \max\{\mathsf{wd}(d_1), |A|, \mathsf{wd}(d_2), |B|\}$$
$$=: 2 \cdot \max\{\mathsf{wd}(d), |A|, |B|\}$$

where we applied the induction hypothesis and Definition 3.29.   □

Combining Theorem 3.9, Proposition 3.19, Proposition 3.30, and Proposition 3.33, we obtain the following.

Theorem 3.34. *Branch width is equivalent to monoidal width in* $\mathsf{Cospan}(\mathsf{UHGraph})_*$. *More precisely, let G be a hypergraph and $g = \emptyset \to G \leftarrow \emptyset$ be the corresponding morphism of* $\mathsf{Cospan}(\mathsf{UHGraph})_*$. *Then, $\frac{1}{2} \cdot \mathsf{bwd}(G) \leq \mathsf{mwd}(g) \leq \mathsf{bwd}(G) + 1$.*

With Theorem 3.9, we obtain:

Corollary 3.35. *Tree width is equivalent to monoidal width in* $\mathsf{Cospan}(\mathsf{UHGraph})_*$.

4. Monoidal width in matrices

We have just seen that instantiating monoidal width in a monoidal category of graphs yields a measure that is equivalent to tree width. Now, we turn our attention to rank width, which is more linear algebraic in flavour as it relies on treating the connectivity of graphs by means of adjacency matrices. Thus, the monoidal category of matrices is a natural example to study first. We relate monoidal width in the category of matrices over the natural numbers, which we introduce in Section 4.1, to their rank (Section 4.2).

The rank of a matrix is the maximum number of its linearly independent rows (or, equivalently, columns). Conveniently, it can be characterised in terms of minimal factorisations.

Lemma 4.1 [PO99]**.** *Let $A \in \mathsf{Mat}_{\mathbb{N}}(m, n)$ be an m by n matrix with entries in the natural numbers. Then $\mathsf{rk}(A) = \min\{k \in \mathbb{N} : \exists B \in \mathsf{Mat}_{\mathbb{N}}(k, n) \ \exists C \in \mathsf{Mat}_{\mathbb{N}}(m, k) \ A = C \cdot B\}$.*

4.1. **The prop of matrices.** The monoidal category $\mathsf{Mat}_{\mathbb{N}}$ of matrices with entries in the natural numbers is a prop whose morphisms from n to m are m by n matrices.

Definition 4.2. $\mathsf{Mat}_{\mathbb{N}}$ is the prop whose morphisms $n \to m$ are m by n matrices with entries in the natural numbers. Composition is the usual product of matrices and the monoidal product is the biproduct $A \otimes B := \left(\begin{smallmatrix} A & \mathbb{0} \\ \mathbb{0} & B \end{smallmatrix} \right)$.

Let us examine matrix decompositions enabled by this algebra. A matrix A can be written as a monoidal product $A = A_1 \otimes A_2$ iff the matrix has blocks A_1 and A_2, i.e. $A = \left(\begin{smallmatrix} A_1 & \mathbb{0} \\ \mathbb{0} & A_2 \end{smallmatrix} \right)$. On the other hand, a composition is related to the rank: the statement of Lemma 4.1 can be read in the category $\mathsf{Mat}_{\mathbb{N}}$ as $\mathsf{rk}(A) = \min\{k \in \mathbb{N} : A = B \mathbin{;_k} C\}$.

Theorem 4.3 [Zan15]**.** *Let Bialg be the prop whose generators and axioms are given in Figure 6. There is an isomorphism of categories* $\mathbf{Mat} : \mathsf{Bialg} \to \mathsf{Mat}_{\mathbb{N}}$.

Every morphism $f : n \to m$ in Bialg corresponds to a matrix $A = \mathbf{Mat}(f) \in \mathsf{Mat}_{\mathbb{N}}(m, n)$: we can read the (i, j)-entry of A off the diagram of f by counting the number of paths from the jth input to the ith output.
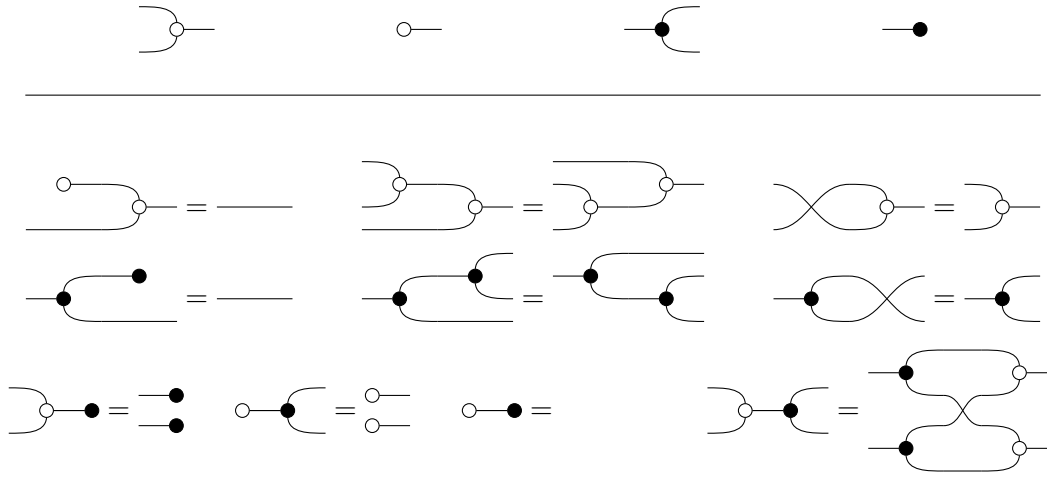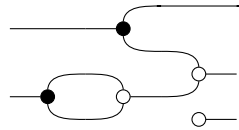
FIGURE 6. Generators and axioms of a bialgebra.

Example 4.4. The matrix $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 2 \\ 0 & 0 \end{smallmatrix}\right) \in \mathsf{Mat}_{\mathbb{N}}(3,2)$ corresponds to

For matrices $A \in \mathsf{Mat}_{\mathbb{N}}(m,n)$, $B \in \mathsf{Mat}_{\mathbb{N}}(m,p)$ and $C \in \mathsf{Mat}_{\mathbb{N}}(l,n)$, we indicate with $(A \mid B) \in \mathsf{Mat}_{\mathbb{N}}(m, n+p)$ and with $\left(\begin{smallmatrix} A \\ C \end{smallmatrix}\right) \in \mathsf{Mat}_{\mathbb{N}}(m+l, n)$ the matrices obtained by concatenating A with B horizontally or with C vertically.

In order to instantiate monoidal width in Bialg, we need to define an appropriate weight function: the natural choice for a prop is to assign weight n to compositions along the object n.

Definition 4.5. The atoms for Bialg are its generators (Figure 6) with the symmetry and identity on 1: $\mathcal{A} = \{ {-}\!\!\blacktriangleleft_1, {-}\!\!\bullet_1, \mathcal{D}{-}_1, \circ{-}_1, \bowtie_{1,1}, \mathbb{1}_1 \}$. The weight function $\mathsf{w}\colon \mathcal{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{Bialg}) \to \mathbb{N}$ has $\mathsf{w}(n) \coloneqq n$, for any $n \in \mathbb{N}$, and $\mathsf{w}(g) \coloneqq \max\{m,n\}$, for $g\colon n \to m \in \mathcal{A}$.

4.2. **Monoidal width of matrices.** We show that the monoidal width of a morphism in the category of matrices Bialg, with the weight function in Definition 4.5, is, up to 1, the maximum rank of its blocks. The overall strategy to prove this result is to first relate monoidal width directly with the rank (Proposition 4.8) and then to improve this bound by prioritising $\otimes$-nodes in a decomposition (Proposition 4.10). Combining these two results leads to Theorem 4.13. The shape of an optimal decomposition is given in Figure 7: a matrix $A = \left(\begin{smallmatrix} A_1 & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & A_2 & \cdots & \mathbb{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{0} & \mathbb{0} & \cdots & A_k \end{smallmatrix}\right)$ can be decomposed as $A = (M_1 \,;\, N_1) \otimes (M_2 \,;\, N_2) \otimes \cdots \otimes (M_k \,;\, N_k)$, where $A_j = M_j \,;\, N_j$ is a rank factorisation as in Lemma 4.1.

The characterisation of the rank of a matrix in Lemma 4.1 hints at some relationship between the monoidal width of a matrix and its rank. In fact, we have Proposition 4.8,
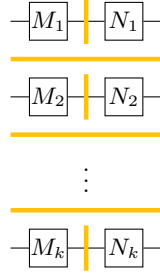
FIGURE 7. Generic shape of an optimal decomposition in Bialg.

which bounds the monoidal width of a matrix with its rank. In order to prove this result, we first need to bound the monoidal width of a matrix with its domain and codomain, which is done in Proposition 4.6.
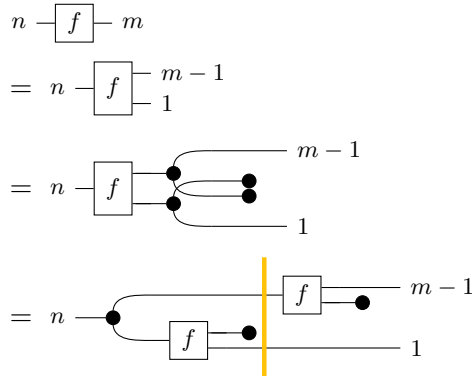
Proposition 4.6. *Let* P *be a cartesian and cocartesian prop. Suppose that* $\mathbb{1}_1, {-\!\!\blacktriangleleft}_1, {\triangleright\!\!-}_1, {-\!\!\bullet}_1,$ ${\circ\!\!-}_1 \in \mathcal{A}$ *and* $\mathsf{w}(\mathbb{1}_1) \leq 1$, $\mathsf{w}({-\!\!\blacktriangleleft}_1) \leq 2$, $\mathsf{w}({\triangleright\!\!-}_1) \leq 2$, $\mathsf{w}({-\!\!\bullet}_1) \leq 1$ *and* $\mathsf{w}({\circ\!\!-}_1) \leq 1$. *Suppose that, for every* $g\colon 1 \to 1$, $\mathsf{mwd}(g) \leq 2$. *Let* $f\colon n \to m$ *be a morphism in* P. *Then* $\mathsf{mwd}(f) \leq \min\{m, n\} + 1$.

Proof. We proceed by induction on $k = \max\{m, n\}$. There are three base cases.

- If $n = 0$, then $f = {\circ\!\!-}_m$ because 0 is initial by hypothesis, and we can compute its width, $\mathsf{mwd}(f) = \mathsf{mwd}(\bigotimes_m {\circ\!\!-}_1) \leq \mathsf{w}({\circ\!\!-}_1) \leq 1 \leq 0 + 1$.
- If $m = 0$, then $f = {-\!\!\bullet}_n$ because 0 is terminal by hypothesis, and we can compute its width, $\mathsf{mwd}(f) = \mathsf{mwd}(\bigotimes_m {-\!\!\bullet}_1) \leq \mathsf{w}({-\!\!\bullet}_1) \leq 1 \leq 0 + 1$.
- If $m = n = 1$, then $\mathsf{mwd}(f) \leq 2 \leq 1 + 1$ by hypothesis.

For the induction steps, suppose that the statement is true for any $f'\colon n' \to m'$ with $\max\{m', n'\} < k = \max\{m, n\}$ and $\min\{m', n'\} \geq 1$. There are three possibilities.

(1) If $0 < n < m = k$, then f can be decomposed as shown below because ${-\!\!\blacktriangleleft}_{n+1}$ is uniform and morphisms are copiable because P is cartesian by hypothesis.

This corresponds to $f = {-\!\!\blacktriangleleft}_n\,; (\mathbb{1}_n \otimes h_1)\,;_{n+1} (h_2 \otimes \mathbb{1}_1)$, where $h_1 := f\,; ({-\!\!\bullet}_{m-1} \otimes \mathbb{1}_1)\colon n \to 1$ and $h_2 := f\,; (\mathbb{1}_{m-1} \otimes {-\!\!\bullet}_1)\colon n \to m - 1$.

Then, $\mathsf{mwd}(f) \leq \max\{\mathsf{mwd}({-\!\!\blacktriangleleft}_n\,; (\mathbb{1}_n \otimes h_1)), n + 1, \mathsf{mwd}(h_2 \otimes \mathbb{1}_1)\}$. So, we want to bound the monoidal width of the two morphisms appearing in the formula above. For the first morphism, we apply the induction hypothesis because $h_1\colon n \to 1$ and $1, n < k$.

For the second morphism, we apply the induction hypothesis because $h_2\colon n \to m-1$ and $n, m-1 < k$.

$\mathsf{mwd}(\text{-}\!\!\mathbb{C}_n \,;\, (\mathbb{1}_n \otimes h_1))$	$\mathsf{mwd}(h_2 \otimes \mathbb{1}_1)$
$\leq$ (by Lemma 2.8)	$=$ (by Definition 2.3)
$\max\{\mathsf{mwd}(h_1), n+1\}$	$\mathsf{mwd}(h_2)$
$\leq$ (by induction hypothesis)	$\leq$ (by induction hypothesis)
$\max\{\min\{n,1\}+1, n+1\}$	$\min\{n, m-1\}+1$
$=$ (because $0 < n$)	$=$ (because $n \leq m-1$)
$n+1$	$n+1$

Then, $\mathsf{mwd}(f) \leq n+1 = \min\{m,n\}+1$ because $n < m$.

(2) If $0 < m < n = k$, we can apply Item 1 to P^{op} with the same assumptions on the set of atoms because P^{op} is also cartesian and cocartesian. We obtain that $\mathsf{mwd}(f) \leq m+1 = \min\{m,n\}+1$ because $m < n$.

(3) If $0 < m = n = k$, f can be decomposed as in Item 1 and, instead of applying the induction hypothesis to bound $\mathsf{mwd}(h_1)$ and $\mathsf{mwd}(h_2)$, one applies Item 2. Then, $\mathsf{mwd}(f) \leq m+1 = \min\{m,n\}+1$ because $m = n$. $\square$

We can apply the former result to Bialg and obtain Proposition 4.8 because the width of 1×1 matrices, which are numbers, is at most 2. This follows from the reasoning in Example 2.5 as we can write every natural number $k\colon 1 \to 1$ as the following composition:

Lemma 4.7. *Let $k\colon 1 \to 1$ in Bialg. Then, $\mathsf{mwd}(k) \leq 2$.*

Proposition 4.8. *Let $f\colon n \to m$ in Bialg. Then, $\mathsf{mwd}f \leq \mathsf{rk}(\mathbf{Mat}f)+1$. Moreover, if f is not $\otimes$-decomposable, i.e. there are no f_1, f_2 both distinct from f s.t. $f = f_1 \otimes f_2$, then $\mathsf{rk}(\mathbf{Mat}f) \leq \mathsf{mwd}f$.*

Proof. We prove the second inequality. Let d be a monoidal decomposition of f. By hypothesis, f is non $\otimes$-decomposable. Then, there are two options.

(1) If the decomposition is just a leaf, $d = (f)$, then f must be an atom. We can check the inequality for all the atoms: $\mathsf{w}(\times) = 2 \geq \mathsf{rk}(\mathbf{Mat}f) = 2$, $\mathsf{w}(\text{-}\!\!\mathbb{C}_1) = \mathsf{w}(\mathbb{C}\!\!\text{-}_1) = 2 \geq \mathsf{rk}(\mathbf{Mat}f) = 1$ or $\mathsf{w}(\text{-}\!\bullet_1) = \mathsf{w}(\circ\!\text{-}_1) = 1 \geq \mathsf{rk}(\mathbf{Mat}f) = 0$. Then, $\mathsf{wd}(d) = \mathsf{w}(f) \geq \mathsf{rk}(\mathbf{Mat}f)$.

(2) If $d = (d_1\text{---}\,;_k\,\text{---}d_2)$, then there are $g\colon n \to k$ and $h\colon k \to m$ such that $f = g\,;h$. By Lemma 4.1, $k \geq \mathsf{rk}(\mathbf{Mat}f)$. Then, $\mathsf{wd}(d) \geq k \geq \mathsf{rk}(\mathbf{Mat}f)$.

We prove the first inequality. By Lemma 4.1, there are $g\colon n \to r$ and $h\colon r \to m$ such that $f = g\,;h$ with $r = \mathsf{rk}(\mathbf{Mat}f)$. Then, $r \leq m, n$ by definition of rank. By Lemma 4.7, we can apply Proposition 4.6 to obtain that $\mathsf{mwd}(g) \leq \min\{n, r\}+1 = r+1$ and $\mathsf{mwd}(h) \leq \min\{m, r\}+1 = r+1$. Then, $\mathsf{mwd}(f) \leq \max\{\mathsf{mwd}(g), r, \mathsf{mwd}(h)\} \leq r+1$. $\square$
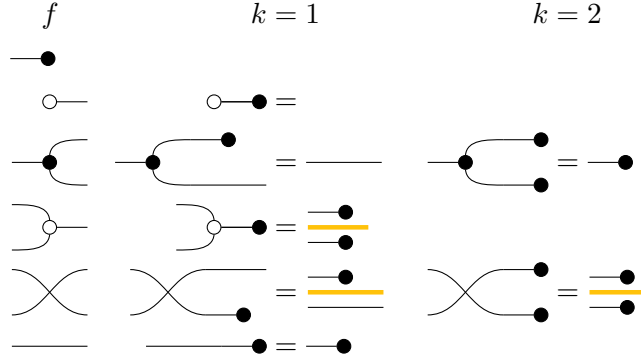
The bounds given by Proposition 4.8 can be improved when we have a $\otimes$-decomposition of a matrix, i.e. we can write $f = f_1 \otimes \ldots \otimes f_k$, to obtain Proposition 4.10. The latter relies on Lemma 4.9, which shows that discarding inputs or outputs cannot increase the monoidal width of a morphism in Bialg.

Lemma 4.9. *Let $f \colon n \to m$ in* Bialg *and $d \in D_f$. Let $f_D := f \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k)$ and $f_Z := (\mathbb{1}_{n-k'} \otimes \circ\text{--}_{k'}) \; ; f$, with $k \leq m$ and $k' \leq n$.*

$$f_D := \; n -\boxed{f}\!\!\to\!\bullet \; m - k \;, \quad f_Z := \; n - k \; \bullet\!\!-\!\!\circ\!-\boxed{f}- \; m \;.$$

Then there are $\mathcal{D}(d) \in D_{f_D}$ and $\mathcal{Z}(d) \in D_{f_Z}$ such that $\mathsf{wd}(\mathcal{D}(d)) \leq \mathsf{wd}(d)$ and $\mathsf{wd}(\mathcal{Z}(d)) \leq \mathsf{wd}(d)$.

Proof. We show the inequality for f_D by induction on the decomposition d. The inequality for f_Z follows from the fact that Bialg coincides with its opposite category. If the decomposition has only one node, $d = (f)$, then f is an atom and we can check these cases by hand in the table below. The first column shows the possibilities for f, while the second and third columns show the decompositions of f_D for $k = 1$ and $k = 2$.

If the decomposition starts with a composition node, $d = (d_1\text{--} \; ; \text{--}d_2)$, then $f = f_1 \; ; f_2$, with d_i monoidal decomposition of f_i.

$$n -\boxed{f}\!\to\!\bullet \; m-k \;\; = \;\; n -\boxed{f_1}-\boxed{f_2}\!\to\!\bullet \; m-k$$

By induction hypothesis, there is a monoidal decomposition $\mathcal{D}(d_2)$ of $f_2 \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k)$ such that $\mathsf{wd}(\mathcal{D}(d_2)) \leq \mathsf{wd}(d_2)$. We use this decomposition to define a decomposition $\mathcal{D}(d) := (d_1\text{--} \; ; \text{--}\mathcal{D}(d_2))$ of f_D. Then, $\mathcal{D}(d)$ is a monoidal decomposition of $f \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k)$ because $f \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k) = f_1 \; ; f_2 \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k)$.

If the decomposition starts with a tensor node, $d = (d_1\text{--} \otimes \text{--}d_2)$, then $f = f_1 \otimes f_2$, with d_i monoidal decomposition of $f_i \colon n_i \to m_i$. There are two possibilities: either $k \leq m_2$ or $k > m_2$. If $k \leq m_2$, then $f \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k) = f_1 \otimes (f_2 \; ; (\mathbb{1}_{m_2-k} \otimes \text{--}\bullet_k))$.

$$n -\boxed{f}\!\to\!\bullet \; m-k \;\; = \;\; \begin{array}{l} n_1 -\boxed{f_1}- \; m_1 \\ n_2 -\boxed{f_2}\!\to\!\bullet \; m_2-k \end{array}$$

By induction hypothesis, there is a monoidal decomposition $\mathcal{D}(d_2)$ of $f_2 \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k)$ such that $\mathsf{wd}(\mathcal{D}(d_2)) \leq \mathsf{wd}(d_2)$. Then, we can use this decomposition to define a decomposition $\mathcal{D}(d) := (d_1\text{--} \otimes \text{--}\mathcal{D}(d_2))$ of f_D. If $k > m_2$, then $f \; ; (\mathbb{1}_{m-k} \otimes \text{--}\bullet_k) = (f_1 \; ; (\mathbb{1}_{m_1-k+m_2} \otimes \text{--}\bullet_{k-m_2})) \otimes (f_2 \; ; \text{--}\bullet_{m_2})$.

$$n -\boxed{f}\!\to\!\bullet \; m-k \;\; = \;\; \begin{array}{l} n_1 -\boxed{f_1}\!\to\!\bullet \; m_1-k+m_2 \\ n_2 -\boxed{f_2}\!\to\!\bullet \end{array}$$

By induction hypothesis, there are monoidal decompositions $\mathcal{D}(d_i)$ of $f_1 \; ; (\mathbb{1}_{m_1-k+m_2} \otimes \text{--}\bullet_{k-m_2})$ and $f_2 \; ; \text{--}\bullet_{m_2}$ such that $\mathsf{wd}(\mathcal{D}(d_i)) \leq \mathsf{wd}(d_i)$. Then, we can use these decompositions to define a monoidal decomposition $\mathcal{D}(d) := (\mathcal{D}(d_1)\text{--} \otimes \text{--}\mathcal{D}(d_2))$ of f_D. $\square$

Proposition 4.10. *Let $f\colon n \to m$ in Bialg and $d' = (d'_1 \mathbin{-\!\!;_k} -d'_2) \in D_f$. Suppose there are f_1 and f_2 such that $f = f_1 \otimes f_2$. Then, there is $d = (d_1 \mathbin{-\!\otimes\!-} d_2) \in D_f$ such that $\mathsf{wd}(d) \le \mathsf{wd}(d')$.*

Proof. By hypothesis, d' is a monoidal decomposition of f. Then, there are g and h such that $f_1 \otimes f_2 = f = g \mathbin{;} h$. By Proposition 4.8, there are monoidal decompositions d_i of f_i with $\mathsf{wd}(d_i) \le r_i + 1$, where $r_i := \mathsf{rk}(\mathbf{Mat}\,f_i)$. By properties of the rank, $r_1 + r_2 = \mathsf{rk}(\mathbf{Mat}\,f)$ and, by Lemma 4.1, $\mathsf{rk}(\mathbf{Mat}\,f) \le k$.

There are two cases: either both ranks are non-zero, or at least one is zero. If $r_i > 0$, then $r_1 + r_2 \ge \max\{r_1, r_2\} + 1$. If there is $r_i = 0$, then $f_i = \mathord{-\!\bullet}\mathbin{;_0}\mathord{\circ\!-}$ and we may assume that $f_1 = \mathord{-\!\bullet}\mathbin{;_0}\mathord{\circ\!-}$. Then, we can express f_2 in terms of g and h.

By Lemma 4.9, $\mathsf{mwd}((\mathbb{1} \otimes \mathord{\circ\!-}) \mathbin{;} g) \le \mathsf{mwd}(g)$ and $\mathsf{mwd}(h \mathbin{;} (\mathbb{1} \otimes \mathord{-\!\bullet})) \le \mathsf{mwd}(h)$. We compute the widths of the decompositions in these two cases.

Case $r_i > 0$ Case $r_1 = 0$

$\mathsf{wd}(d')$ $\mathsf{wd}(d')$

$= \max\{\mathsf{wd}(d'_1), k, \mathsf{wd}(d'_2)\}$ $= \max\{\mathsf{wd}(d'_1), k, \mathsf{wd}(d'_2)\}$

$\ge k$ $\ge \max\{\mathsf{mwd}(g), k, \mathsf{mwd}(h)\}$

$\ge \mathsf{rk}(\mathbf{Mat}\,f)$ $\ge \max\{\mathsf{mwd}((\mathbb{1} \otimes \mathord{\circ\!-}) \mathbin{;} g), k, \mathsf{mwd}(h \mathbin{;} (\mathbb{1} \otimes \mathord{-\!\bullet}))\}$

$= r_1 + r_2$ $\ge \mathsf{mwd}(f_2)$

$\ge \max\{r_1, r_2\} + 1$ $= \mathsf{wd}(d_2)$

$\ge \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2)\}$ $= \mathsf{wd}(d)$

$= \mathsf{wd}(d)$ $\square$

We summarise Proposition 4.10 and Proposition 4.8 in Corollary 4.11.

Corollary 4.11. *Let $f = f_1 \otimes \ldots \otimes f_k$ in Bialg. Then, $\mathsf{mwd}(f) \le \max_{i=1,\ldots,k} \mathsf{rk}(\mathbf{Mat}(f_i)) + 1$. Moreover, if f_i are not $\otimes$-decomposable, then $\max_{i=1,\ldots,k} \mathsf{rk}(\mathbf{Mat}(f_i)) \le \mathsf{mwd}\,f$.*
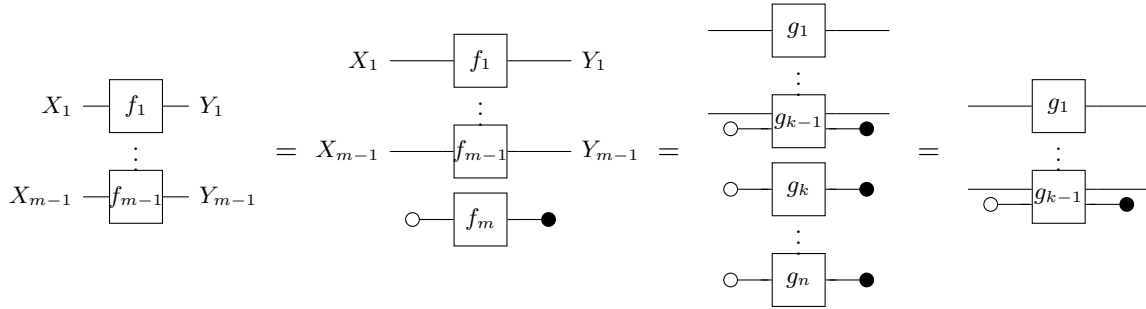
Proof. By Proposition 4.10 there is a decomposition of f of the form $d = (d_1 \mathbin{-\!\otimes\!-} \cdots (d_{k-1} \mathbin{-\!\otimes} \mathbin{-}d_k))$, where we can choose d_i to be a minimal decomposition of f_i. Then, $\mathsf{mwd}(f) \le \mathsf{wd}(d) = \max_{i=1,\ldots,k} \mathsf{wd}(d_i)$. By Proposition 4.8, $\mathsf{wd}(d_i) \le r_i + 1$. Then, $\mathsf{mwd}(f) \le \max\{r_1, \ldots, r_k\} + 1$. Moreover, if f_i are not $\otimes$-decomposable, Proposition 4.8 gives also a lower bound on their monoidal width: $\mathsf{rk}(\mathbf{Mat}(f_i)) \le \mathsf{mwd}\,f_i$; and we obtain that $\max_{i=1,\ldots,k} \mathsf{rk}(\mathbf{Mat}(f_i)) \le \mathsf{mwd}\,f$. $\square$

The results so far show a way to construct efficient decompositions given a $\otimes$-decomposition of the matrix. However, we do not know whether $\otimes$-decompositions are unique. Proposition 4.12 shows that every morphism in Bialg has a unique $\otimes$-decomposition.
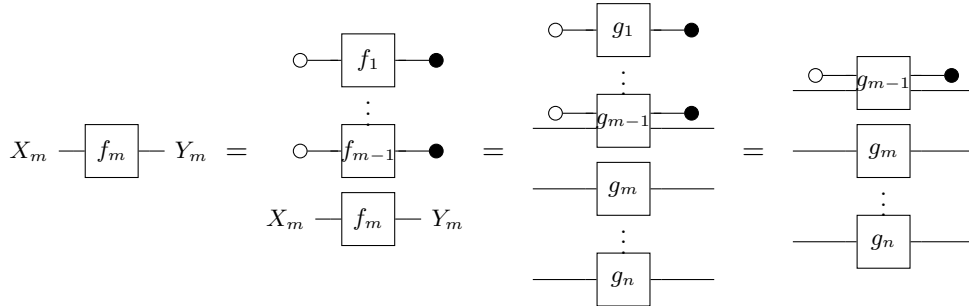
Proposition 4.12. *Let C be a monoidal category whose monoidal unit 0 is both initial and terminal, and whose objects are a unique factorisation monoid. Let f be a morphism in C. Then f has a unique $\otimes$-decomposition.*

Proof. Suppose $f = f_1 \otimes \cdots \otimes f_m = g_1 \otimes \cdots \otimes g_n$ with $f_i \colon X_i \to Y_i$ and $g_j \colon Z_j \to W_j$ non $\otimes$-decomposables. Suppose $m \leq n$ and proceed by induction on m. If $m = 0$, then $f = \mathbb{1}_0$ and $g_i = \mathbb{1}_0$ for every $i = 1, \ldots, n$ because 0 is initial and terminal.

Suppose that $\bar{f} := f_1 \otimes \ldots \otimes f_{m-1}$ has a unique $\otimes$-decomposition. Let $A_1 \otimes \ldots \otimes A_\alpha$ and $B_1 \otimes \ldots \otimes B_\beta$ be the unique $\otimes$-decompositions of $X_1 \otimes \ldots \otimes X_m = Z_1 \otimes \ldots \otimes Z_n$ and $Y_1 \otimes \ldots \otimes Y_m = W_1 \otimes \ldots \otimes W_n$, respectively. Then, there are $x \leq \alpha$ and $y \leq \beta$ such that $A_1 \otimes \ldots \otimes A_x = X_1 \otimes \ldots \otimes X_{m-1}$ and $B_1 \otimes \ldots \otimes B_y = Y_1 \otimes \ldots \otimes Y_{m-1}$. Then, we can rewrite $\bar{f}$ in terms of g_is:

By induction hypothesis, $\bar{f}$ has a unique $\otimes$-decomposition, thus it must be that $k = m - 1$, for every $i < m - 1$ $f_i = g_i$ and $f_{m-1} = (\mathbb{1} \otimes \circ\!\!-) \, ; g_k \, ; (\mathbb{1} \otimes -\!\!\bullet)$. Then, we can express f_m in terms of $g_m, \ldots, g_n$:

By hypothesis, f_m is not $\otimes$-decomposable and $m \leq n$. Thus, $n = m$, $f_{m-1} = g_{m-1}$ and $f_m = g_m$. $\qquad\square$

Our main result in this section follows from Corollary 4.11 and Proposition 4.12, which can be applied to Bialg because 0 is both terminal and initial, and the objects, being a free monoid, are a unique factorisation monoid.

Theorem 4.13. *Let $f = f_1 \otimes \ldots \otimes f_k$ be a morphism in* Bialg *and its unique $\otimes$-decomposition given by Proposition 4.12, with $r_i = \mathsf{rk}(\mathbf{Mat}(f_i))$. Then $\max\{r_1, \ldots, r_k\} \leq \mathsf{mwd}(f) \leq \max\{r_1, \ldots, r_k\} + 1$.*

Note that the identity matrix has monoidal width 1 and twice the identity matrix has monoidal width 2, attaining both the upper and lower bounds for the monoidal width of a matrix.

5. A MONOIDAL ALGEBRA FOR RANK WIDTH

After having studied monoidal width in the monoidal category of matrices, we are ready to introduce the second monoidal category of "open graphs", which relies on matrices to encode the connectivity of graphs. In this setting, we capture rank width: we show that instantiating monoidal width in this monoidal category of graphs is equivalent to rank width.

   After recalling rank width in Section 5.1, we define the intermediate notion of inductive rank decomposition in Section 5.2, and show its equivalence to that of rank decomposition. As for branch decompositions, adding this intermediate step allows a clearer presentation of the correspondence between rank decompositions and monoidal decompositions. Section 5.3 recalls the categorical algebra of graphs with boundaries [CS15, DLHS21]. Finally, Section 5.4 contains the main result of the present section, which relates inductive rank decompositions, and thus rank decompositions, with monoidal decompositions.

   Rank decompositions were originally defined for undirected graphs [OS06]. This motivates us to consider graphs rather than hypergraphs as in Section 3. As mentioned in Definition 3.3, a finite undirected graph is a finite undirected hypergraph with hyperedge size 2. More explicitly,

Definition 5.1. A *graph* $G = (V, E)$ is given by a finite set of vertices V, a finite set of edges E and an adjacency function $\mathsf{ends}\colon E \to \wp_{\leq 2}(V)$, where $\wp_{\leq 2}(V)$ indicates the set of subsets of V with at most two elements. The same information recorded in the function ends can be encoded in an equivalence class of matrices, an *adjacency matrix* $[G]$: the sum of the entries (i, j) and (j, i) of this matrix records the number of edges between vertex i and vertex j; two adjacency matrices are equivalent when they encode the same graph, i.e. $[G] = [H]$ iff $G + G^\top = H + H^\top$.

5.1. **Background: rank width.** Intuitively, rank width measures the amount of information needed to construct a graph by adding edges to a discrete graph. Constructing a clique requires little information: we add an edge between any two vertices. This is reflected in the fact that cliques have rank width 1.

   Rank width relies on rank decompositions. In analogy with branch decompositions, a rank decomposition records in a tree a way of iteratively partitioning the vertices of a graph.

Definition 5.2 [OS06]**.** A *rank decomposition* (Y, r) of a graph G is given by a subcubic tree Y together with a bijection $r\colon \mathsf{leaves}(Y) \to \mathsf{vertices}(G)$.

   Each edge b in the tree Y determines a splitting of the graph: it determines a two partition of the leaves of Y, which, through r, determines a 2-partition $\{A_b, B_b\}$ of the vertices of G. This corresponds to a splitting of the graph G into two subgraphs G_1 and G_2. Intuitively, the order of an edge b is the amount of information required to recover G by joining G_1 and G_2. Given the partition $\{A_b, B_b\}$ of the vertices of G, we can record the edges in G beween A_b and B_b in a matrix X_b. This means that, if $v_i \in A_b$ and $v_j \in B_b$, the entry (i, j) of the matrix X_b is the number of edges between v_i and v_j.

Definition 5.3 (Order of an edge)**.** Let (Y, r) be a rank decomposition of a graph G. Let b be an edge of Y. The order of b is the rank of the matrix associated to it: $\mathsf{ord}(b) := \mathsf{rk}(X_b)$.

   Note that the order of the two sets in the partition does not matter as the rank is invariant to transposition. The width of a rank decomposition is the maximum order of the edges of the tree and the rank width of a graph is the width of its cheapest decomposition.

Definition 5.4 (Rank width). Given a rank decomposition (Y, r) of a graph G, define its width as $\mathsf{wd}(Y, r) \coloneqq \max_{b \in \mathsf{edges}(Y)} \mathsf{ord}(b)$. The *rank width* of G is given by the min-max formula:
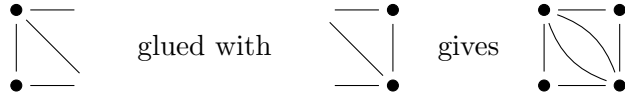
$$\mathsf{rwd}(G) \coloneqq \min_{(Y, r)} \mathsf{wd}(Y, r).$$

5.2. **Graphs with dangling edges and inductive definition.** We introduce graphs with dangling edges and inductive rank decomposition of them. These decompositions are an intermediate notion between rank decompositions and monoidal decompositions. Similarly to the definition of inductive branch decomposition (Section 3.2), they add to rank decompositions the algebraic flavour of monoidal decompositions by using the inductive data type of binary trees to encode a decomposition.
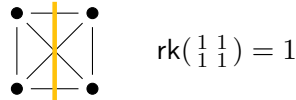
Intuitively, a graph with dangling edges is a graph equipped with some extra edges that connect some vertices in the graph to some boundary ports. This allows us to combine graphs with dangling edges by connecting some of their dangling edges. Thus, the equivalence between rank decompositions and inductive rank decompositions formalises the intuition that a rank decomposition encodes a way of dividing a graph into smaller subgraphs by "cutting" along some edges.

Definition 5.5. A *graph with dangling edges* $\Gamma = ([G], B)$ is given by an adjacency matrix $G \in \mathsf{Mat}_{\mathbb{N}}(k, k)$ that records the connectivity of the graph and a matrix $B \in \mathsf{Mat}_{\mathbb{N}}(k, n)$ that records the "dangling edges" connected to n boundary ports. We will sometimes write $G \in \mathsf{adjacency}(\Gamma)$ and $B = \mathsf{sources}(\Gamma)$.

Example 5.6. Two graphs with the same ports, as illustrated below, can be "glued" together:

A rank decomposition is, intuitively, a recipe for decomposing a graph into its single-vertex subgraphs by cutting along its edges. The cost of each cut is given by the rank of the adjacency matrix that represents it.

Decompositions are elements of a tree data type, with nodes carrying subgraphs Γ' of the ambient graph Γ. In the following Γ' ranges over the non-empty subgraphs of Γ: $T_\Gamma ::= (\Gamma') \mid (T_\Gamma \text{---} \Gamma' \text{---} T_\Gamma)$. Given $T \in T_\Gamma$, the label function λ takes a decomposition and returns the graph with dangling edges at the root: $\lambda(T_1 \text{---} \Gamma \text{---} T_2) \coloneqq \Gamma$ and $\lambda((\Gamma)) \coloneqq \Gamma$.

The conditions in the definition of inductive rank decomposition ensure that, by glueing Γ_1 and Γ_2 together, we get Γ back.

Definition 5.7. Let $\Gamma = ([G], B)$ be a graph with dangling edges, where $G \in \mathsf{Mat}_{\mathbb{N}}(k, k)$ and $B \in \mathsf{Mat}_{\mathbb{N}}(k, n)$. An *inductive rank decomposition* of Γ is $T \in T_\Gamma$ where either: Γ is empty and $T = ()$; or Γ has one vertex and $T = (\Gamma)$; or $T = (T_1 \text{---} \Gamma \text{---} T_2)$ and $T_i \in T_{\Gamma_i}$ are inductive rank decompositions of subgraphs $\Gamma_i = ([G_i], B_i)$ of Γ such that:

• The vertices are partitioned in two, $[G] = \left[\begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix} \right]$;

- The dangling edges are those to the original boundary and to the other subgraph, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$, where $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$.

We will sometimes write $\Gamma_i = \lambda(T_i)$, $G_i = \mathsf{adjacency}(\Gamma_i)$ and $B_i = \mathsf{sources}(\Gamma_i)$. We can always assume that the rows of G and B are ordered like the leaves of T so that we can actually split B horizontally to get A_1 and A_2.

Remark 5.8. The perspective on rank width and branch width given by their inductive definitions emphasises an operational difference between them: a branch decompositon gives a recipe to construct a graph from its one-edge subgraphs by identifying some of their vertices; on the other hand, a rank decomposition gives a recipe to construct a graph from its one-vertex components by connecting some of their "dangling" edges.

Definition 5.9. Let $T = (T_1\text{---}\Gamma\text{---}T_2)$ be an inductive rank decomposition of $\Gamma = ([G], B)$, with T_i possibly both empty. Define the *width* of T inductively: if T is empty, $\mathsf{wd}(()) := 0$; otherwise, $\mathsf{wd}(T) := \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rk}(B)\}$. Expanding this expression, we obtain

$$\mathsf{wd}(T) = \max_{T' \text{ full subtree of } T} \mathsf{rk}(\mathsf{sources}(\lambda(T'))).$$

The *inductive rank width* of Γ is defined by the min-max formula $\mathsf{irwd}(\Gamma) := \min_T \mathsf{wd}(T)$.

We show that the inductive rank width of $\Gamma = ([G], B)$ is the same as the rank width of G, up to the rank of the boundary matrix B.

Before proving the upper bound for inductive rank width, we need a technical lemma that relates the width of a graph with that of its subgraphs and allows us to compute it "globally".

Lemma 5.10. *Let T be an inductive rank decomposition of $\Gamma = ([G], B)$. Let T' be a full subtree of T and $\Gamma' := \lambda(T')$ with $\Gamma' = ([G'], B')$. The adjacency matrix of Γ can be written as $[G] = \left[\begin{pmatrix} G_L & C_L & C \\ \mathbb{0} & G' & C_R \\ \mathbb{0} & \mathbb{0} & G_R \end{pmatrix} \right]$ and its boundary as $B = \begin{pmatrix} A_L \\ A' \\ A_R \end{pmatrix}$. Then, $\mathsf{rk}(B') = \mathsf{rk}(A' \mid C_L^\top \mid C_R)$.*

Proof. Proceed by induction on the decomposition tree T. If it is just a leaf, $T = (\Gamma)$, then Γ has at most one vertex, and $\Gamma' = \emptyset$ or $\Gamma' = \Gamma$. In both cases, the desired equality is true.

If $T = (T_1\text{---}\Gamma\text{---}T_2)$, then, by the definition of inductive rank decomposition, $\lambda(T_i) = \Gamma_i = ([G_i], B_i)$ with $[G] = \left[\begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix} \right]$, $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$. Suppose that $T' \leq T_1$. Then, we can write $[G_1] = \left[\begin{pmatrix} G_L & C_L & D' \\ \mathbb{0} & G' & D_R \\ \mathbb{0} & \mathbb{0} & H_R \end{pmatrix} \right]$, $A_1 = \begin{pmatrix} A_L \\ A' \\ F_R \end{pmatrix}$ and $C = \begin{pmatrix} E_L \\ E' \\ E_R \end{pmatrix}$. It follows that $B_1 = \begin{pmatrix} A_L & E_L \\ A' & E' \\ F_R & E_R \end{pmatrix}$ and $C_R = (D_R \mid E')$. By induction hypothesis, $\mathsf{rk}(B') = \mathsf{rk}(A' \mid E' \mid C_L^\top \mid D_R)$. The rank is invariant to permuting the order of columns, thus $\mathsf{rk}(B') = \mathsf{rk}(A' \mid C_L^\top \mid D_R \mid E') = \mathsf{rk}(A' \mid C_L^\top \mid C_R)$. We proceed analogously if $T' \leq T_2$. $\square$

The above result allows us to relate the width of rank decompositions, which is computed "globally", to the width of inductive rank decompositions, which is computed "locally", with the following bound.

Proposition 5.11. *Let $\Gamma = ([G], B)$ be a graph with dangling edges and (Y, r) be a rank decomposition of G. Then, there is an inductive rank decomposition $\mathcal{I}(Y, r)$ of Γ such that $\mathsf{wd}(\mathcal{I}(Y, r)) \leq \mathsf{wd}(Y, r) + \mathsf{rk}(B)$.*

Proof. Proceed by induction on the number of edges of the decomposition tree Y to construct an inductive decomposition tree T in which every non-trivial full subtree T' has a corresponding edge b' in the tree Y. Suppose Y has no edges, then either $G = \emptyset$ or G has one vertex. In either case, we define an inductive rank decomposition with just a leaf labelled with Γ, $\mathcal{I}(Y, r) := (\Gamma)$. We compute its width by definition: $\mathsf{wd}(\mathcal{I}(Y, r)) := \mathsf{rk}(B) \leq \mathsf{wd}(Y, r) + \mathsf{rk}(B)$.

If the decomposition tree has at least an edge, then it is composed of two subcubic subtrees, $Y = Y_1 \overset{b}{\relbar} Y_2$. Let $V_i := r(\mathsf{leaves}(Y_i))$ be the set of vertices associated to Y_i and $G_i := G[V_i]$ be the subgraph of G induced by the set of vertices V_i. By induction hypothesis, there are inductive rank decompositions T_i of $\Gamma_i = ([G_i], B_i)$ in which every full subtree T' has an associated edge b'. Associate the edge b to both T_1 and T_2 so that every subtree of T has an associated edge in Y. We can use these decompositions to define an inductive rank decomposition $T = (T_1 \relbar \Gamma \relbar T_2)$ of Γ. Let T' be a full subtree of T corresponding to $\Gamma' = ([G'], B')$. By Lemma 5.10, we can compute the rank of its boundary matrix $\mathsf{rk}(B') = \mathsf{rk}(A' \mid C_L^\top \mid C_R)$, where A', C_L and C_R are defined as in the statement of Lemma 5.10. The matrix A' contains some of the rows of B, then its rank is bounded by the rank of B and we obtain $\mathsf{rk}(B') \leq \mathsf{rk}(B) + \mathsf{rk}(C_L^\top \mid C_R)$. The matrix $(C_L^\top \mid C_R)$ records the edges between the vertices in G' and the vertices in the rest of G, which, by definition, are the edges that determine $\mathsf{ord}(b')$. This means that the rank of this matrix is the order of the edge b': $\mathsf{rk}(C_L^\top \mid C_R) = \mathsf{ord}(b')$. With these observations, we can compute the width of T.

$$
\begin{aligned}
&\mathsf{wd}(T) \\
&= \max_{T' \leq T} \mathsf{rk}(B') \\
&= \max_{T' \leq T} \mathsf{rk}(A' \mid C_L^\top \mid C_R) \\
&\leq \max_{T' \leq T} \mathsf{rk}(C_L^\top \mid C_R) + \mathsf{rk}(B) \\
&= \max_{b \in \mathsf{edges}(Y)} \mathsf{ord}(b) + \mathsf{rk}(B) \\
&=: \mathsf{wd}(Y, r) + \mathsf{rk}(B) \qquad\qquad \square
\end{aligned}
$$

Proposition 5.12. *Let T be an inductive rank decomposition of $\Gamma = ([G], B)$ with $G \in \mathsf{Mat}_\mathbb{N}(k, k)$ and $B \in \mathsf{Mat}_\mathbb{N}(k, n)$. Then, there is a rank decomposition $\mathcal{I}^\dagger(T)$ of G such that $\mathsf{wd}(\mathcal{I}^\dagger(T)) \leq \mathsf{wd}(T)$.*

Proof. A binary tree is, in particular, a subcubic tree. Then, the rank decomposition corresponding to an inductive rank decomposition T can be defined by its underlying unlabelled tree Y. The corresponding bijection $r \colon \mathsf{leaves}(Y) \to \mathsf{vertices}(G)$ between the leaves of Y and the vertices of G can be defined by the labels of the leaves in T: the label of a leaf l of T is a subgraph of Γ with one vertex v_l and these subgraphs need to give Γ when composed together. Then, the leaves of T, which are the leaves of Y, are in bijection with the vertices of G: there is a bijection $r \colon \mathsf{leaves}(Y) \to \mathsf{vertices}(G)$ such that $r(l) := v_l$. Then, (Y, r) is a branch decomposition of G and we can define $\mathcal{I}^\dagger(T) := (Y, r)$.

By construction, the edges of Y are the same as the edges of T so we can compute the order of the edges in Y from the labellings of the nodes in T. Consider an edge b in Y and consider its endpoints in T: let $\{v, v_b\} = \mathsf{ends}(b)$ with v parent of v_b in T. The order of b is related to the rank of the boundary of the subtree T_b of T with root in v_b. Let $\lambda(T_b) = \Gamma_b = ([G_b], B_b)$ be the subgraph of Γ identified by T_b. We can express the adjacency

and boundary matrices of Γ in terms of those of Γ_b:

$$[G] = \left[\begin{pmatrix} G_L & C_L & C \\ \mathbb{0} & G_b & C_R \\ \mathbb{0} & \mathbb{0} & G_R \end{pmatrix} \right] \qquad \text{and} \qquad B = \begin{pmatrix} A_L \\ A' \\ A_R \end{pmatrix}.$$

By Lemma 5.10, the boundary rank of Γ_b can be computed by $\mathsf{rk}(B_b) = \mathsf{rk}(A' \mid C_L^\top \mid C_R)$. By definition, the order of the edge b is $\mathsf{ord}(b) := \mathsf{rk}(C_L^\top \mid C_R)$, and we can bound it with the boundary rank of Γ_b: $\mathsf{rk}(B_b) \geq \mathsf{ord}(b)$. These observations allow us to bound the width of the rank decomposition Y that corresponds to T.

$$
\begin{aligned}
\mathsf{wd}(Y, r) \\
&:= \max_{b \in \mathsf{edges}(Y)} \mathsf{ord}(b) \\
&\leq \max_{b \in \mathsf{edges}(Y)} \mathsf{rk}(B_b) \\
&\leq \max_{T' \leq T} \mathsf{rk}(\mathsf{sources}(\lambda(T'))) \\
&=: \mathsf{wd}(T) \qquad\qquad\qquad\qquad \square
\end{aligned}
$$

Combining Proposition 5.11 and Proposition 5.12 we obtain:

Proposition 5.13. *Inductive rank width is equivalent to rank width.*

5.3. **A prop of graphs.** Here we recall the algebra of graphs with boundaries and its diagrammatic syntax [DLHS21]. Graphs with boundaries are graphs together with some extra "dangling" edges that connect the graph to the left and right boundaries. They compose by connecting edges that share a common boundary. All the information about connectivity is handled with matrices.
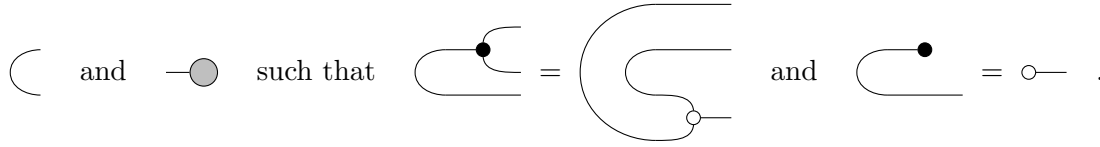
Remark 5.14. The categorical algebra of graphs with boundaries is a natural choice for capturing rank width because it emphasises the operation of splitting a graph into parts that share some *edges*. This contrasts with the algebra of cospans of graphs (Section 3.3), in which graphs are split into subgraphs that share some *vertices*. The difference in the operation that is emphasised by these two algebras reflects the difference between rank width and tree or branch width pointed out in Remark 5.8.

Definition 5.15 [DLHS21]. A *graph with boundaries* $g \colon n \to m$ is a tuple $g = ([G], L, R, P, [F])$ of an adjacency matrix $[G]$ of a graph on k vertices, with $G \in \mathsf{Mat}_\mathbb{N}(k, k)$; matrices $L \in \mathsf{Mat}_\mathbb{N}(k, n)$ and $R \in \mathsf{Mat}_\mathbb{N}(k, m)$ that record the connectivity of the vertices with the left and right boundary; a matrix $P \in \mathsf{Mat}_\mathbb{N}(m, n)$ that records the passing wires from the left boundary to the right one; and a matrix $F \in \mathsf{Mat}_\mathbb{N}(m, m)$ that records the wires from the right boundary to itself. Graphs with boundaries are taken up to an equivalence making the order of the vertices immaterial. Let $g, g' \colon n \to m$ on k vertices, with $g = ([G], L, R, P, [F])$ and $g' = ([G'], L', R', P, [F])$. The graphs g and g' are considered equal iff there is a permutation matrix $\sigma \in \mathsf{Mat}_\mathbb{N}(k, k)$ such that $g' = ([\sigma G \sigma^\top], \sigma L, \sigma R, P, [F])$.

Graphs with boundaries can be composed sequentially and in parallel [DLHS21], forming a symmetric monoidal category MGraph.

The prop Grph provides a convenient syntax for graphs with boundaries. It is obtained by adding a cup and a vertex generators to the prop of matrices Bialg (Figure 6).

Definition 5.16 [CS15]**.** The prop of graphs Grph is obtained by adding to Bialg the generators $\cup\colon 0 \to 2$ and $\mathsf{v}\colon 1 \to 0$ with the equations below.

These equations mean, in particular, that the cup transposes matrices (Figure 8, left) and that we can express the equivalence relation of adjacency matrices as in Definition 5.1: $[G] = [H]$ iff $G + G^\top = H + H^\top$ (Figure 8, right).
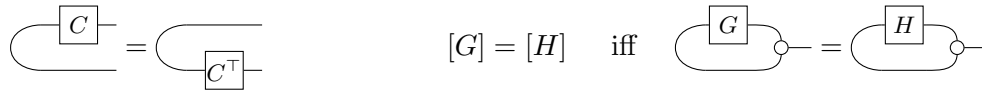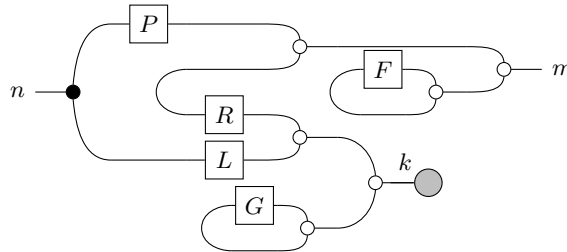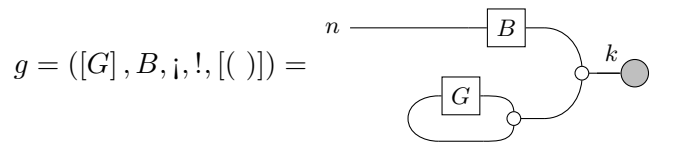
Figure 8. Adding the cup.

Proposition 5.17 [DLHS21, Theorem 23]**.** *The prop of graphs* Grph *is isomorphic to the prop* MGraph.

Proposition 5.17 means that the morphisms in Grph can be written in the following normal form

The prop Grph is more expressive than graphs with dangling edges (Definition 5.5): its morphisms can have edges between the boundaries as well. In fact, graphs with dangling edges can be seen as morphisms $n \to 0$ in Grph.

Example 5.18. A graph with dangling edges $\Gamma = ([G], B)$ can be represented as a morphism in Grph

$$g = ([G], B, \text{¡}, !, [(\ )]) =$$

where $!\colon n \to 0$ and $\text{¡}\colon 0 \to k$ are the unique maps to and from the terminal and initial object 0. We can now formalise the intuition of glueing graphs with dangling edges as explained in Example 5.6. The two graphs there correspond to g_1 and g_2 below left and middle. Their glueing is obtained by precomposing their monoidal product with a cup, i.e. $\cup_2 \,; (g_1 \otimes g_2)$,

as shown below right.

Definition 5.19. Let the set of *atomic morphisms* $\mathcal{A}$ be the set of all the morphisms of Grph. The *weight function* $\mathsf{w} \colon \mathcal{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{Grph}) \to \mathbb{N}$ is defined, on objects n, as $\mathsf{w}(n) \coloneqq n$; and, on morphisms $g \in \mathcal{A}$, as $\mathsf{w}(g) \coloneqq k$, where k is the number of vertices of g.

Note that, the monoidal width of g is bounded by the number k of its vertices, thus we could take as atoms all the morphisms with at most one vertex and the results would not change.

5.4. **Rank width as monoidal width.** We show that monoidal width in the prop Grph, with the weight function given in Definition 5.19, is equivalent to rank width. We do this by bounding monoidal width by above with twice rank width and by below with half of rank width (Theorem 5.26). We prove these bounds by defining maps from inductive rank decompositions to monoidal decompositions that preserve the width (Proposition 5.23), and vice versa (Proposition 5.25).

The upper bound (Proposition 5.23) is established by associating to each inductive rank decomposition a suitable monoidal decomposition. This mapping is defined inductively, given the inductive nature of both these structures. Given an inductive rank decomposition of a graph Γ, we can construct a decomposition of its corresponding morphism g as shown by the first equality in Figure 9. However, this decomposition is not optimal as it cuts along
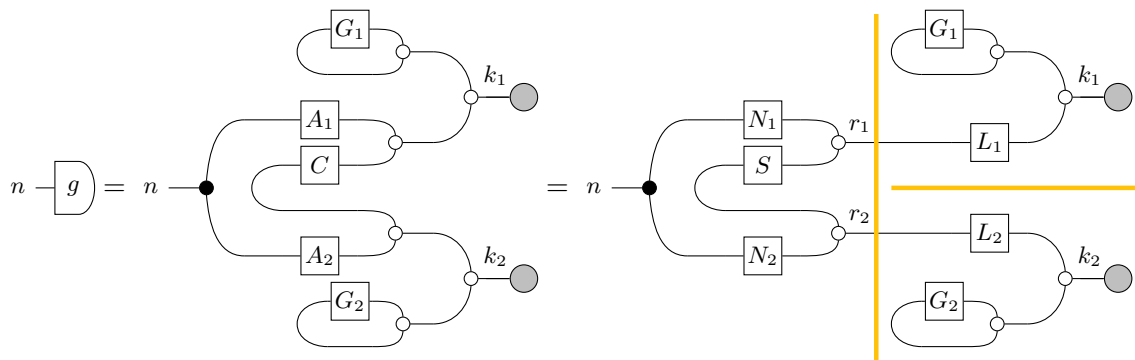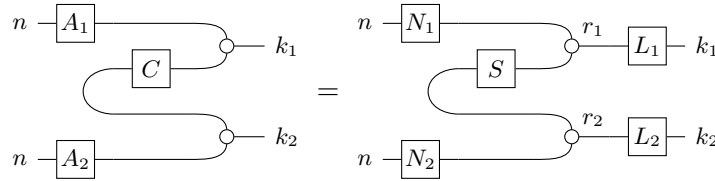

FIGURE 9. First step of a monoidal decomposition given by an inductive rank decomposition

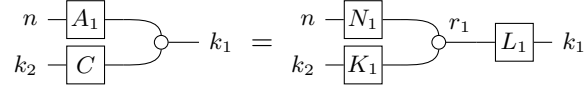the number of vertices $k_1 + k_2$. But we can do better thanks to Lemma 5.21, which shows

that we can cut along the ranks, $r_1 = \mathsf{rk}(A_1 \mid C)$ and $r_2 = \mathsf{rk}(A_2 \mid C^\top)$, of the boundaries of the induced subgraphs to obtain the second equality in Figure 9. First, recall some facts about ranks.

Remark 5.20. By Lemma 4.1, the rank of a composition of two matrices is bounded by their ranks: $\mathsf{rk}(A \cdot B) \leq \min\{\mathsf{rk}(A), \mathsf{rk}(B)\}$. If, moreover, B has full rank, then $\mathsf{rk}(A \cdot B) = \mathsf{rk}(A)$.
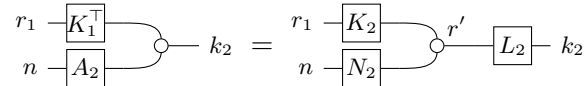
Lemma 5.21. *Let* $A_i \in \mathsf{Mat}_\mathbb{N}(k_i, n)$, *for* $i = 1, 2$, *and* $C \in \mathsf{Mat}_\mathbb{N}(k_1, k_2)$. *Then, there are rank decompositions of* $(A_1 \mid C)$ *and* $(A_2 \mid C^\top)$ *of the form* $(A_1 \mid C) = L_1 \cdot (N_1 \mid S \cdot L_2^\top)$, *and* $(A_2 \mid C^\top) = L_2 \cdot (N_2 \mid S^\top \cdot L_1^\top)$. *This ensures that we can decompose the diagram below on the left-hand-side as the one on the right-hand-side, where* $r_1 = \mathsf{rk}(A_1 \mid C)$ *and* $r_2 = \mathsf{rk}(A_2 \mid C^\top)$.

Proof. Let $r_1 = \mathsf{rk}(A_1 \mid C)$ and $r_2 = \mathsf{rk}(A_2 \mid C^\top)$. We start by factoring $(A_1 \mid C)$ into $L_1 \cdot (N_1 \mid K_1)$,

where $L_1 \in \mathsf{Mat}_\mathbb{N}(k_1, r_1)$, $N_1 \in \mathsf{Mat}_\mathbb{N}(r_1, n)$ and $K_1 \in \mathsf{Mat}_\mathbb{N}(r_1, k_2)$. Then, we proceed with factoring $(A_2 \mid K_1^\top)$ and we show that $\mathsf{rk}(A_2 \mid K_1^\top) = \mathsf{rk}(A_2 \mid C^\top)$. Let $L_2 \cdot (N_2 \mid K_2)$ be a rank factorisation of $(A_2 \mid K_1^\top)$,

with $L_2 \in \mathsf{Mat}_\mathbb{N}(k_2, r')$, $N_2 \in \mathsf{Mat}_\mathbb{N}(r', n)$ and $K_2 \in \mathsf{Mat}_\mathbb{N}(r', k_1)$. We show that $r' = r_2$. By the first factorisation, we obtain that $C = L_1 \cdot K_1$, and

$$(A_2 \mid C^\top) = (A_2 \mid K_1^\top \cdot L_1^\top) = (A_2 \mid K_1^\top) \cdot \begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & L_1^\top \end{pmatrix}.$$

Then, $r' = r_2$ because L_1 and, consequently, $\begin{pmatrix} \mathbb{1} & \mathbb{0} \\ \mathbb{0} & L_1^\top \end{pmatrix}$ have full rank. By letting $S = K_2^\top$, we obtain the desired factorisation. $\qquad\square$

Once we have performed the cuts in Figure 9 on the right, we have changed the boundaries of the induced subgraphs. This means that we cannot apply the inductive hypothesis right away, but we need to transform first the inductive rank decompositions of the old subgraphs into decompositions of the new ones, as shown in Lemma 5.22. More explicitly, when M has full rank, if we have an inductive rank decomposition of $\Gamma = ([G], B \cdot M)$, which corresponds to g below left, we can obtain one of $\Gamma' = ([G], B \cdot M')$, which corresponds to g' below right, of at most the same width.

Lemma 5.22. *Let T be an inductive rank decomposition of $\Gamma = ([G], B \cdot M)$, with M that has full rank. Then, there is an inductive rank decomposition T' of $\Gamma' = ([G], B \cdot M')$ such that $\mathsf{wd}(T) \leq \mathsf{wd}(T')$ and such that T and T' have the same underlying tree structure. If, moreover, M' has full rank, then $\mathsf{wd}(T) = \mathsf{wd}(T')$.*

Proof. Proceed by induction on the decomposition tree T. If the tree T is just a leaf with label Γ, then we define the corresponding tree to be just a leaf with label Γ': $T' := (\Gamma')$. Clearly, T and T' have the same underlying tree structure. By Remark 5.20 and the fact that M has full rank, we can relate their widths: $\mathsf{wd}(T') := \mathsf{rk}(B \cdot M') \leq \mathsf{rk}(B) = \mathsf{rk}(B \cdot M) =: \mathsf{wd}(T)$. If, moreover, M' has full rank, the inequality becomes an equality and $\mathsf{wd}(T') = \mathsf{wd}(T)$.

If $T = (T_1 \text{---} \Gamma \text{---} T_2)$, then the adjacency and boundary matrices of Γ can be expressed in terms of those of its subgraphs $\Gamma_i := \lambda_i(T_i) = ([G_i], D_i)$, by definition of inductive rank decomposition: $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B \cdot M = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \cdot M = \begin{pmatrix} A_1 \cdot M \\ A_2 \cdot M \end{pmatrix}$, with $D_1 = (A_1 \cdot M \mid C)$ and $D_2 = (A_2 \cdot M \mid C^\top)$. The boundary matrices D_i of the subgraphs Γ_i can also be expressed as a composition with a full-rank matrix: $D_1 = (A_1 \cdot M \mid C) = (A_1 \mid C) \cdot \begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_2} \end{pmatrix}$ and $D_2 = (A_2 \cdot M \mid C^\top) = (A_2 \mid C^\top) \cdot \begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_1} \end{pmatrix}$. The matrices $\begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_i} \end{pmatrix}$ have full rank because all their blocks do. Let $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$. By induction hypothesis, there are inductive rank decompositions T_1' and T_2' of $\Gamma_1' = ([G_1], B_1 \cdot \begin{pmatrix} M' & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_2} \end{pmatrix})$ and $\Gamma_2' = ([G_2], B_2 \cdot \begin{pmatrix} M' & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_1} \end{pmatrix})$ with the same underlying tree structure as T_1 and T_2, respectively. Moreover, their width is bounded, $\mathsf{wd}(T_i') \leq \mathsf{wd}(T_i)$, and if, additionally, M' has full rank, $\mathsf{wd}(T_i') = \mathsf{wd}(T_i)$. Then, we can use these decompositions to define an inductive rank decomposition $T' := (T_1' \text{---} \Gamma' \text{---} T_2')$ of Γ' because its adjacency and boundary matrices can be expressed in terms of those of Γ_i' as in the definition of inductive rank decomposition: $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B_1 \cdot \begin{pmatrix} M' & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_2} \end{pmatrix} = (A_1 \cdot M' \mid C)$ and $B_2 \cdot \begin{pmatrix} M' & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_1} \end{pmatrix} = (A_2 \cdot M' \mid C^\top)$. Applying the induction hypothesis and Remark 5.20, we compute the width of this decomposition.

$$\begin{aligned}
& \mathsf{wd}(T') \\
& := \max\{\mathsf{rk}(B \cdot M'), \mathsf{wd}(T_1'), \mathsf{wd}(T_2')\} \\
& \leq \max\{\mathsf{rk}(B), \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
& = \max\{\mathsf{rk}(B \cdot M), \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
& =: \mathsf{wd}(T)
\end{aligned}$$

If, moreover, M' has full rank, the inequality becomes an equality and $\mathsf{wd}(T') = \mathsf{wd}(T)$. $\square$

With the above ingredients, we can show that rank width bounds monoidal width from above.

Proposition 5.23. *Let $\Gamma = ([G], B)$ be a graph with dangling edges and $g: n \to 0$ be the morphism in Grph corresponding to Γ. Let T be an inductive rank decomposition of Γ. Then, there is a monoidal decomposition $\mathcal{R}^\dagger(T)$ of g such that $\mathsf{wd}(\mathcal{R}^\dagger(T)) \leq 2 \cdot \mathsf{wd}(T)$.*

Proof. Proceed by induction on the decomposition tree T. If it is empty, then G must also be empty, $\mathcal{R}^\dagger(T) = ()$ and we are done. If the decomposition tree consists of just one leaf with label Γ, then Γ must have one vertex, we can define $\mathcal{R}^\dagger(T) := (g)$ to also be just a leaf, and bound its width $\mathsf{wd}(T) := \mathsf{rk}(G) = \mathsf{wd}(\mathcal{R}^\dagger(T))$.

If $T = (T_1 \text{—} \Gamma \text{—} T_2)$, then we can relate the adjacency and boundary matrices of Γ to those of $\Gamma_i := \lambda(T_i) = ([G_i], B_i)$, by definition of inductive rank decomposition: $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$. By Lemma 5.21, there are rank decompositions of $(A_1 \mid C)$ and $(A_2 \mid C^\top)$ of the form: $(A_1 \mid C) = L_1 \cdot (N_1 \mid S \cdot L_2^\top)$; and $(A_2 \mid C^\top) = L_2 \cdot (N_2 \mid S^\top \cdot L_1^\top)$. This means that we can write g as

with $r_i = \mathsf{rk}(B_i)$. Then, $B_i = L_i \cdot M_i$ with M_i that has full rank r_i. By taking $M' = \mathbb{1}$ in Lemma 5.22, there is an inductive rank decomposition T_i' of $\Gamma_i' = ([G_i], L_i)$, with the same underlying binary tree as T_i, such that $\mathsf{wd}(T_i) = \mathsf{wd}(T_i')$. Let $g_i : r_i \to 0$ be the morphisms in Grph corresponding to Γ_i' and let $b : n \to r_1 + r_2$ be defined as

By induction hypothesis, there are monoidal decompositions $\mathcal{R}^\dagger(T_1')$ and $\mathcal{R}^\dagger(T_2')$ of g_1 and g_2 of bounded width: $\mathsf{wd}(\mathcal{R}^\dagger(T_i')) \le 2 \cdot \mathsf{wd}(T_i') = 2 \cdot \mathsf{wd}(T_i)$. Then, $g = b \,;_{r_1+r_2} (g_1 \otimes g_2)$ and $\mathcal{R}^\dagger(T) := (b \text{—} ;_{r_1+r_2} \text{—} (\mathcal{R}^\dagger(T_1') \text{—} \otimes \text{—} \mathcal{R}^\dagger(T_2')))$ is a monoidal decomposition of g. Its width can be computed.

$$
\begin{aligned}
&\mathsf{wd}(\mathcal{R}^\dagger(T)) \\
&:= \max\{\mathsf{w}(b), \mathsf{w}(r_1 + r_2), \mathsf{wd}(\mathcal{R}^\dagger(T_1')), \mathsf{wd}(\mathcal{R}^\dagger(T_2'))\} \\
&\le \max\{\mathsf{w}(b), \mathsf{w}(r_1 + r_2), 2 \cdot \mathsf{wd}(T_1'), 2 \cdot \mathsf{wd}(T_2')\} \\
&= \max\{\mathsf{w}(b), r_1 + r_2, 2 \cdot \mathsf{wd}(T_1), 2 \cdot \mathsf{wd}(T_2)\} \\
&\le 2 \cdot \max\{r_1, r_2, \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
&=: 2 \cdot \mathsf{wd}(T) \hspace{6cm} \square
\end{aligned}
$$

Proving the lower bound is similarly involved and follows a similar proof structure. From a monoidal decomposition we construct inductively an inductive rank decomposition of bounded width. The inductive step relative to composition nodes is the most involved and needs two additional lemmas, which allow us to transform inductive rank decompositions of the induced subgraphs into ones of two subgraphs that satisfy the conditions of Definition 5.7.

Applying the inductive hypothesis gives us an inductive rank decomposition of $\Gamma = ([G], (L \mid R))$, which is associated to g below left, and we need to construct one of $\Gamma' :=$

$([G + L \cdot F \cdot L^\top], (L \mid R + L \cdot (F + F^\top) \cdot P^\top))$, which is associated to $f \,;\, g$ below right, of at most the same width.

Lemma 5.24. *Let T be an inductive rank decomposition of $\Gamma = ([G], (L \mid R))$, with $G \in \mathsf{Mat}_\mathbb{N}(k, k)$, $L \in \mathsf{Mat}_\mathbb{N}(k, j)$ and $R \in \mathsf{Mat}_\mathbb{N}(k, m)$. Let $F \in \mathsf{Mat}_\mathbb{N}(j, j)$, $P \in \mathsf{Mat}_\mathbb{N}(m, j)$ and define $\Gamma' := ([G + L \cdot F \cdot L^\top], (L \mid R + L \cdot (F + F^\top) \cdot P^\top))$. Then, there is an inductive rank decomposition T' of Γ' such that $\mathsf{wd}(T') \leq \mathsf{wd}(T)$.*

Proof. Note that we can factor the boundary matrix of Γ' as $(L \mid R + L \cdot (F + F^\top) \cdot P^\top) = (L \mid R) \cdot \begin{pmatrix} \mathbb{1}_j & (F + F^\top) \cdot P^\top \\ \mathbb{0} & \mathbb{1}_m \end{pmatrix}$. Then, we can bound its rank, $\mathsf{rk}(L \mid R + L \cdot (F + F^\top) \cdot P^\top) \leq \mathsf{rk}(L \mid R)$.

Proceed by induction on the decomposition tree T. If it is just a leaf with label Γ, then Γ has one vertex and we can define a decomposition for Γ' to be also just a leaf: $T' := (\Gamma')$. We can bound its width with the width of T: $\mathsf{wd}(T') := \mathsf{rk}(L \mid R + L \cdot (F + F^\top) \cdot P^\top) \leq \mathsf{rk}(L \mid R) =: \mathsf{wd}(T)$.

If $T = (T_1 \text{---} \Gamma \text{---} T_2)$, then there are two subgraphs $\Gamma_1 = ([G_1], (L_1 \mid R_1 \mid C))$ and $\Gamma_2 = ([G_2], (L_2 \mid R_2 \mid C))$ such that T_i is an inductive rank decomposition of Γ_i, and we can relate the adjacency and boundary matrices of Γ to those of Γ_1 and Γ_2, by definition of inductive rank decomposition: $[G] = \left[\begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}\right]$ and $(L \mid R) = \begin{pmatrix} L_1 & R_1 \\ L_2 & R_2 \end{pmatrix}$. Similarly, we express the adjacency and boundary matrices of Γ' in terms of the same components: $[G + L \cdot F \cdot L^\top] = \left[\begin{pmatrix} G_1 + L_1 \cdot F \cdot L_1^\top & C + L_1 \cdot (F + F^\top) \cdot L_2^\top \\ \mathbb{0} & G_2 + L_2 \cdot F \cdot L_2^\top \end{pmatrix}\right]$ and $(L \mid R + L \cdot (F + F^\top) \cdot P^\top) = \begin{pmatrix} L_1 & R_1 + L_1 \cdot (F + F^\top) \cdot P^\top \\ L_2 & R_2 + L_2 \cdot (F + F^\top) \cdot P^\top \end{pmatrix}$. We use these decompositions to define two subgraphs of Γ' and apply the induction hypothesis to them.

$$\Gamma_1' := ([G_1 + L_1 \cdot F \cdot L_1^\top], (L_1 \mid R_1 + L_1 \cdot (F + F^\top) \cdot P^\top \mid C + L_1 \cdot (F + F^\top) \cdot L_2^\top))$$

$$= ([G_1 + L_1 \cdot F \cdot L_1^\top], (L_1 \mid (R_1 \mid C) + L_1 \cdot (F + F^\top) \cdot (P^\top \mid L_2^\top)))$$

and

$$\Gamma_2' := ([G_2 + L_2 \cdot F \cdot L_2^\top], (L_2 \mid R_2 + L_2 \cdot (F + F^\top) \cdot P^\top \mid C^\top + L_2 \cdot (F + F^\top) \cdot L_1^\top))$$

$$= ([G_2 + L_2 \cdot F \cdot L_2^\top], (L_2 \mid (R_2 \mid C^\top) + L_2 \cdot (F + F^\top) \cdot (P^\top \mid L_1^\top)))$$

By induction, we have inductive rank decompositions T_i' of Γ_i' such that $\mathsf{wd}(T_i') \leq \mathsf{wd}(T_i)$. We defined Γ_i' so that $T' := (T_1'\!-\!\Gamma'\!-\!T_2')$ would be an inductive rank decomposition of Γ'. We can bound its width as desired.
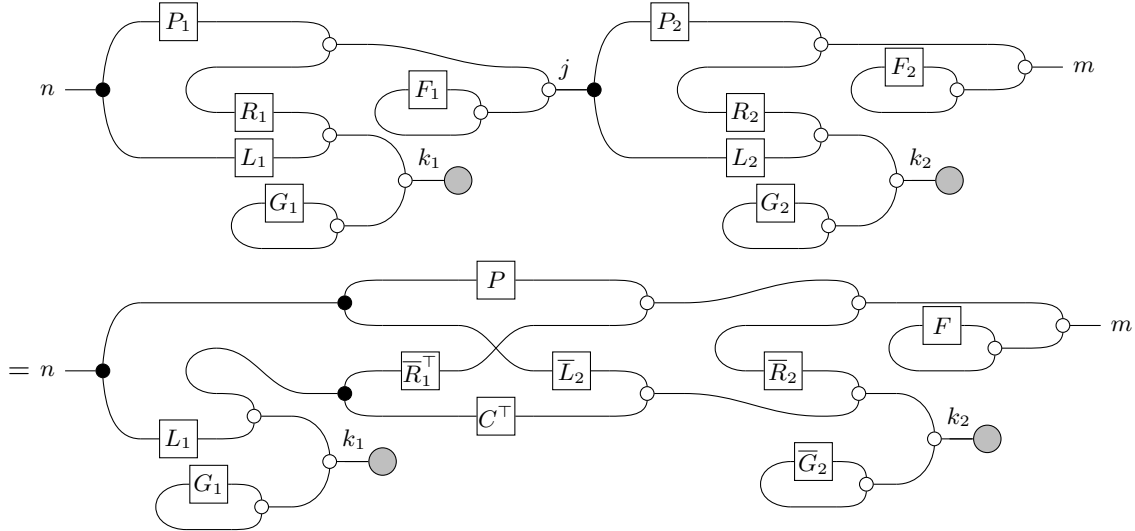
$$\mathsf{wd}(T')$$
$$:= \max\{\mathsf{wd}(T_1'), \mathsf{wd}(T_2'), \mathsf{rk}(L \mid R + L \cdot (F + F^\top) \cdot P^\top)\}$$
$$\leq \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rk}(L \mid R + L \cdot (F + F^\top) \cdot P^\top)\}$$
$$\leq \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rk}(L \mid R)\}$$
$$=: \mathsf{wd}(T) \qquad\qquad \square$$

In order to obtain the subgraphs of the desired shape we need to add some extra connections to the boundaries. This can be done thanks to Lemma 5.22, by taking $M = \mathbb{1}$. We are finally able to prove the lower bound for monoidal width.

Proposition 5.25. *Let* $g = ([G], L, R, P, [F])$ *in* Grph *and* $d \in D_g$. *Let* $\Gamma = ([G], (L \mid R))$. *Then, there is an inductive rank decomposition* $\mathcal{R}(d)$ *of* Γ *s.t.* $\mathsf{wd}(\mathcal{R}(d)) \leq 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rk}(L), \mathsf{rk}(R)\}$.

Proof. Proceed by induction on the decomposition tree d. If it is just a leaf with label g, then its width is defined to be the number k of vertices of g, $\mathsf{wd}(d) := k$. Pick any inductive rank decomposition of Γ and define $\mathcal{R}(d) := T$. Surely, $\mathsf{wd}(T) \leq k =: \mathsf{wd}(d)$

If $d = (d_1\!-\!;_j\!-\!d_2)$, then g is the composition of two morphisms: $g = g_1 \,;\, g_2$, with $g_i = ([G_i], L_i, R_i, P_i, [F_i])$. Given the partition of the vertices determined by g_1 and g_2, we can decompose g in another way, by writing $[G] = \left[\left(\begin{smallmatrix} \overline{G}_1 & C \\ \mathbb{0} & \overline{G}_2 \end{smallmatrix}\right)\right]$ and $B = (L \mid R) = \left(\begin{smallmatrix} \overline{L}_1 & \overline{R}_1 \\ \overline{L}_2 & \overline{R}_2 \end{smallmatrix}\right)$. Then, we have that $\overline{G}_1 = G_1$, $\overline{L}_1 = L_1$, $P = P_2 \cdot P_1$, $C = R_1 \cdot L_2^\top$, $\overline{R}_1 = R_1 \cdot P_2^\top$, $\overline{L}_2 = L_2 \cdot P_1$, $\overline{R}_2 = R_2 + L_2 \cdot (F_1 + F_1^\top) \cdot P_2^\top$, $\overline{G}_2 = G_2 + L_2 \cdot F_1 \cdot L_2^\top$, and $F = F_2 + P_2 \cdot F_1 \cdot P_2^\top$. This corresponds to the following diagrammatic rewriting using the equations of Grph.

We define $\overline{B}_1 := (\overline{L}_1 \mid \overline{R}_1 \mid C)$ and $\overline{B}_2 := (\overline{L}_2 \mid \overline{R}_2 \mid C^\top)$. In order to build an inductive rank decomposition of Γ, we need rank decompositions of $\overline{\Gamma}_i = ([\overline{G}_i], \overline{B}_i)$. We obtain these in three steps. Firstly, we apply induction to obtain inductive rank decompositions $\mathcal{R}(d_i)$ of $\Gamma_i = ([G_i], (L_i \mid R_i))$ such that $\mathsf{wd}(\mathcal{R}(d_i)) \leq 2 \cdot \max\{\mathsf{wd}(d_i), \mathsf{rk}(L_i), \mathsf{rk}(R_i)\}$. Secondly, we apply

Lemma 5.24 to obtain an inductive rank decomposition T_2' of $\Gamma_2' = (\left[G_2 + L_2 \cdot F_1 \cdot L_2^\top\right], (L_2 \mid R_2 + L_2 \cdot (F_1 + F_1^\top) \cdot P_2^\top))$ such that $\mathsf{wd}(T_2') \leq \mathsf{wd}(\mathcal{R}(d_2))$. Lastly, we observe that $(\overline{R}_1 \mid C) = R_1 \cdot (P_2^\top \mid L_2^\top)$ and $(\overline{L}_2 \mid C^\top) = L_2 \cdot (P_1 \mid R_1^\top)$. Then we obtain that $\overline{B}_1 = (L_1 \mid R_1) \cdot \left(\begin{smallmatrix} \mathbb{1}_n & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & P_2^\top & L_2^\top \end{smallmatrix}\right)$ and $\overline{B}_2 = (L_2 \mid R_2 + L_2 \cdot (F_1 + F_1^\top) \cdot P_2^\top) \cdot \left(\begin{smallmatrix} P_1 & \mathbb{0} & R_1^\top \\ \mathbb{0} & \mathbb{1}_m & \mathbb{0} \end{smallmatrix}\right)$, and we can apply Lemma 5.22, with $M = \mathbb{1}$, to get inductive rank decompositions T_i of $\overline{\Gamma}_i$ such that $\mathsf{wd}(T_1) \leq \mathsf{wd}(\mathcal{R}(d_1))$ and $\mathsf{wd}(T_2) \leq \mathsf{wd}(T_2') \leq \mathsf{wd}(\mathcal{R}(d_2))$. If $k_1, k_2 > 0$, then we define $\mathcal{R}(d) := (T_1\text{---}\Gamma\text{---}T_2)$, which is an inductive rank decomposition of Γ because $\overline{\Gamma}_i$ satisfy the two conditions in Definition 5.7. If $k_1 = 0$, then $\Gamma = \overline{\Gamma}_2$ and we can define $\mathcal{R}(d) := T_2$. Similarly, if $k_2 = 0$, then $\Gamma = \overline{\Gamma}_1$ and we can define $\mathcal{R}(d) := T_1$. In any case, we can compute the width of $\mathcal{R}(d)$ (if $k_i = 0$ then $T_i = ()$ and $\mathsf{wd}(T_i) = 0$) using the inductive hypothesis, Lemma 5.24, Lemma 5.22, the fact that $\mathsf{rk}(L) \geq \mathsf{rk}(L_1)$, $\mathsf{rk}(R) \geq \mathsf{rk}(R_2)$ and $j \geq \mathsf{rk}(R_1), \mathsf{rk}(L_2)$ because $R_1 \colon j \to k_1$ and $L_2 \colon j \to k_2$.

$\mathsf{wd}(T)$

$:= \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rk}(L \mid R)\}$

$\leq \max\{\mathsf{wd}(\mathcal{R}(d_1)), \mathsf{wd}(T_2'), \mathsf{rk}(L \mid R)\}$

$\leq \max\{\mathsf{wd}(\mathcal{R}(d_1)), \mathsf{wd}(\mathcal{R}(d_2)), \mathsf{rk}(L \mid R)\}$

$\leq \max\{\mathsf{wd}(\mathcal{R}(d_1)), \mathsf{wd}(\mathcal{R}(d_2)), \mathsf{rk}(L) + \mathsf{rk}(R)\}$

$\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot \mathsf{rk}(L_1), 2 \cdot \mathsf{rk}(R_1), 2 \cdot \mathsf{wd}(d_2), 2 \cdot \mathsf{rk}(L_2), 2 \cdot \mathsf{rk}(R_2), \mathsf{rk}(L) + \mathsf{rk}(R)\}$

$\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{rk}(L_1), \mathsf{rk}(R_1), \mathsf{wd}(d_2), \mathsf{rk}(L_2), \mathsf{rk}(R_2), \mathsf{rk}(L), \mathsf{rk}(R)\}$

$\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2), j, \mathsf{rk}(L), \mathsf{rk}(R)\}$

$=: 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rk}(L), \mathsf{rk}(R)\}$

If $d = (d_1\text{---}\otimes\text{---}d_2)$, then g is the monoidal product of two morphisms: $g = g_1 \otimes g_2$, with $g_i = ([G_i], L_i, R_i, P_i, [F_i]) \colon n_i \to m_i$. By exlicitly computing the monoidal product, we obtain that $[G] = \left[\left(\begin{smallmatrix} G_1 & \mathbb{0} \\ \mathbb{0} & G_2 \end{smallmatrix}\right)\right]$, $L = \left(\begin{smallmatrix} L_1 & \mathbb{0} \\ \mathbb{0} & L_2 \end{smallmatrix}\right)$, $R = \left(\begin{smallmatrix} R_1 & \mathbb{0} \\ \mathbb{0} & R_2 \end{smallmatrix}\right)$, $P = \left(\begin{smallmatrix} P_1 & \mathbb{0} \\ \mathbb{0} & P_2 \end{smallmatrix}\right)$ and $F = \left(\begin{smallmatrix} F_1 & \mathbb{0} \\ \mathbb{0} & F_2 \end{smallmatrix}\right)$. By induction, we have inductive rank decompositions $\mathcal{R}(d_i)$ of $\Gamma_i := ([G_i], B_i)$, where $B_i = (L_i \mid R_i)$, of bounded width: $\mathsf{wd}(\mathcal{R}(d_i)) \leq 2 \cdot \max\{\mathsf{wd}(d_i), \mathsf{rk}(L_i), \mathsf{rk}(R_i)\}$. Let $\overline{B}_1 := (L_1 \mid \mathbb{0}_{n_2} \mid R_1 \mid \mathbb{0}_{m_2} \mid \mathbb{0}_{k_2}) = B_1 \cdot \left(\begin{smallmatrix} \mathbb{1}_{n_1} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{m_1} & \mathbb{0} & \mathbb{0} \end{smallmatrix}\right)$ and $\overline{B}_2 := (\mathbb{0}_{n_1} \mid L_2 \mid \mathbb{0}_{m_1} \mid R_2 \mid \mathbb{0}_{k_1}) = B_2 \cdot \left(\begin{smallmatrix} \mathbb{0} & \mathbb{1}_{n_2} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{m_2} & \mathbb{0} \end{smallmatrix}\right)$. By taking $M = \mathbb{1}$ in Lemma 5.22, we can obtain inductive rank decompositions T_i of $\overline{\Gamma}_i := ([G_i], \overline{B}_i)$ such that $\mathsf{wd}(T_i) \leq \mathsf{wd}(\mathcal{R}(d_i))$. If $k_1, k_2 > 0$, then we define $\mathcal{R}(d) := (T_1\text{---}\Gamma\text{---}T_2)$, which is an inductive rank decomposition of Γ because $\overline{\Gamma}_i$ satisfy the two conditions in Definition 5.7. If $k_1 = 0$, then $\Gamma = \overline{\Gamma}_2$ and we can define $\mathcal{R}(d) := T_2$. Similarly, if $k_2 = 0$, then $\Gamma = \overline{\Gamma}_1$ and we can define $\mathcal{R}(d) := T_1$. In any case, we can compute the width of $\mathcal{R}(d)$ (if $k_i = 0$ then $T_i = ()$ and $\mathsf{wd}(T_i) = 0$) using the inductive hypothesis and Lemma 5.22.

$\mathsf{wd}(T)$

$:= \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rk}(L \mid R)\}$

$\leq \max\{\mathsf{wd}(\mathcal{R}(d_1)), \mathsf{wd}(\mathcal{R}(d_2)), \mathsf{rk}(L \mid R)\}$

$\leq \max\{\mathsf{wd}(\mathcal{R}(d_1)), \mathsf{wd}(\mathcal{R}(d_2)), \mathsf{rk}(L) + \mathsf{rk}(R)\}$

$\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot \mathsf{rk}(L_1), 2 \cdot \mathsf{rk}(R_1), 2 \cdot \mathsf{wd}(d_2), 2 \cdot \mathsf{rk}(L_2), 2 \cdot \mathsf{rk}(R_2), \mathsf{rk}(L) + \mathsf{rk}(R)\}$

$$\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{rk}(L_1), \mathsf{rk}(R_1), \mathsf{wd}(d_2), \mathsf{rk}(L_2), \mathsf{rk}(R_2), \mathsf{rk}(L), \mathsf{rk}(R)\}$$
$$\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2), \mathsf{rk}(L), \mathsf{rk}(R)\}$$
$$=: 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rk}(L), \mathsf{rk}(R)\} \qquad \square$$

From Proposition 5.23, Proposition 5.25 and Proposition 5.13, we obtain the main result of this section.

Theorem 5.26. *Let G be a graph and let $g = ([G], i, i, (\ ), [(\ )])$ be the corresponding morphism in* Grph. *Then, $\frac{1}{2} \cdot \mathsf{rwd}(G) \leq \mathsf{mwd}(g) \leq 2 \cdot \mathsf{rwd}(G)$.*

6. Conclusion and future work

We defined monoidal width for measuring the complexity of morphisms in monoidal categories. The concrete examples that we aimed to capture are tree width and rank width. In fact, we have shown that, by choosing suitable categorical algebras, monoidal width is equivalent to these widths. We have also related monoidal width to the rank of matrices over the natural numbers.

Our future goal is to leverage the generality of monoidal categories to study other examples outside the graph theory literature. In the same way Courcelle's theorem gives fixed-parameter tractability of a class of problems on graphs with parameter tree width or rank width, we aim to obtain fixed-parameter tractability of a class of problems on morphisms of monoidal categories with parameter monoidal width. This result would rely on Feferman-Vaught-Mostowski type theorems specific to the operations of a particular monoidal category C or particular class of monoidal categories, which would ensure that the problems at hand respect the compositional structure of these categories.

Conjecture. Computing a compositional problem on the set of morphisms $\mathsf{C}_k(X, Y)$ with k-bounded monoidal width with a compositional algorithm is linear in w. Explicitly, computing the solution on $f \in \mathsf{C}_k(X, Y)$ takes $\mathcal{O}(c(k) \cdot \mathsf{w}(f))$, for some more than exponential function $c \colon \mathbb{N} \to \mathbb{N}$.

References

[ADW17]   Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in finite model theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017. `doi:10.1109/LICS.2017.8005129`.

[AM21]   Samson Abramsky and Dan Marsden. Comonadic semantics for guarded fragments. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021. `doi:10.1109/LICS52264.2021.9470594`.

[AS21]   Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation*, 31(6):1390–1428, 2021. `doi:10.1093/logcom/exab048`.

[BB73]   Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973. `doi:10.1016/0097-3165(73)90016-2`.

[BBFK11]   Christoph Blume, HJ Sander Bruggink, Martin Friedrich, and Barbara König. Treewidth, pathwidth and cospan decompositions. *Electronic Communications of the EASST*, 41, 2011. `doi:10.14279/tuj.eceasst.41.643`.

[BC87]   Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20(1):83–127, 1987. `doi:10.1007/BF01692060`.

[BK08]   Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008. `doi:10.1093/comjnl/bxm037`.

[BK21]   Benjamin Merlin Bumpus and Zoltan A Kocsis. Spined categories: generalizing tree-width beyond graphs, 2021. `arXiv:2104.01841`.

[BKM23]   Benjamin Merlin Bumpus, Zoltan Kocsis, and Jade Edenstar Master. Structured decompositions: Structural and algorithmic compositionality, 2023. `arXiv:2207.06091`.

[Bod92]   Hans L Bodlaender. A tourist guide through treewidth. *Technical report*, 1992.

[BS21]   Guillaume Boisseau and Paweł Sobociński. String diagrammatic electrical circuit theory. In Kohei Kishida, editor, Proceedings of the Fourth International Conference on *Applied Category Theory,* Cambridge, United Kingdom, 12-16th July 2021, volume 372 of *Electronic Proceedings in Theoretical Computer Science*, pages 178–191. Open Publishing Association, 2021. `doi:10.4204/EPTCS.372.13`.

[BSZ21]   Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A survey of compositional signal flow theory. In Michael Goedicke, Erich J. Neuhold, and Kai Rannenberg, editors, *Advancing Research in Information and Communication Technology - IFIP's Exciting First 60+ Years, Views from the Technical Committees and Working Groups*, volume 600 of *IFIP Advances in Information and Communication Technology*, pages 29–56. Springer, 2021. `doi:10.1007/978-3-030-81701-5\_2`.

[Bum21]   Benjamin Merlin Bumpus. *Generalizing graph decompositions*. PhD thesis, University of Glasgow, 2021.

[CD21]   Adam Ó Conghaile and Anuj Dawar. Game comonads & generalised quantifiers. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.CSL.2021.16`.

[CJ19]   Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. `doi:10.1017/S0960129518000488`.

[CK17]   Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press, 2017. `doi:10.1017/9781316219317`.

[CK22]   Cole Comfort and Aleks Kissinger. A graphical calculus for lagrangian relations. In Kohei Kishida, editor, Proceedings of the Fourth International Conference on *Applied Category Theory,* Cambridge, United Kingdom, 12-16th July 2021, volume 372 of *Electronic Proceedings in Theoretical Computer Science*, pages 338–351. Open Publishing Association, 2022. `doi:10.4204/EPTCS.372.24`.

[Cou90]   Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. `doi:10.1016/0890-5401(90)90043-H`.

[Cou92]   Bruno Courcelle. The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *RAIRO-Theoretical Informatics and Applications*, 26(3):257–286, 1992. `doi:10.1051/ita/1992260302571`.

[CS15]   Apiwat Chantawibul and Paweł Sobociński. Towards compositional graph theory. *Electronic Notes in Theoretical Computer Science*, 319:121–136, 2015. `doi:10.1016/j.entcs.2015.12.009`.

[DKPvdW20] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum*, 4:279, 2020. `doi:10.22331/q-2020-06-04-279`.

[DLHS21]   Elena Di Lavore, Jules Hedges, and Paweł Sobociński. Compositional modelling of network games. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:24, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.CSL.2021.30`.

[DLS21]   Elena Di Lavore and Paweł Sobociński. Monoidal Width: Unifying Tree Width, Path Width and Branch Width, 2021. `arXiv:2202.07582`.

[DLS22]   Elena Di Lavore and Paweł Sobociński. Monoidal Width: Capturing Rank Width, to appear in ACT 2022. `arXiv:2202.07582`.

[Fon15] Brendan Fong. Decorated cospans. *Theory and Applications of Categories*, 30(33):1096–1120, 2015.

[Fri20] Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020.

[FS18] Brendan Fong and David I Spivak. Seven sketches in compositionality: An invitation to applied category theory, 2018. arXiv:1803.05316.

[GH97] Fabio Gadducci and Reiko Heckel. An inductive view of graph transformation. In Francesco Parisi Presicce, editor, *International Workshop on Algebraic Development Techniques*, pages 223–237. Springer, 1997. doi:10.1007/3-540-64299-4_36.

[GHWZ18] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. Compositional game theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 472–481. ACM, 2018. doi:10.1145/3209108.3209165.

[GJL17] Dan R. Ghica, Achim Jung, and Aliaume Lopez. Diagrammatic semantics for digital circuits. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden*, volume 82 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CSL.2017.24.

[Hal76] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1):171–186, 1976. doi:10.1007/BF01917434.

[JS91] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in mathematics*, 88(1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.

[Lac04] Stephen Lack. Composing props. *Theory and Applications of Categories*, 13(9):147–163, 2004.

[Mac78] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1978. doi:10.1007/978-1-4757-4721-8.

[Mas22] Jade Master. How to compose shortest paths, 2022. arXiv:2205.15306.

[MS22] Yoàv Montacute and Nihil Shah. The pebble-relation comonad in finite model theory. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11. ACM, 2022. doi:10.1145/3531130.3533335.

[OS06] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

[PO99] R Piziak and PL Odell. Full rank factorization of matrices. *Mathematics magazine*, 72(3):193–201, 1999. doi:10.1080/0025570X.1999.11996730.

[PRS88] Pavel Pudlák, Vojtěch Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988. doi:10.1007/bf00279952.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

[RS91] Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. doi:10.1016/0095-8956(91)90061-n.

[RSS14] Julian Rathke, Paweł Sobociński, and Owen Stephens. Compositional reachability in Petri nets. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *International Workshop on Reachability Problems, Lecture Notes in Computer Science*, pages 230–243. Springer, 2014. doi:10.1007/978-3-319-11439-2_18.

[RSW05] Robert Rosebrugh, Nicoletta Sabadini, and Robert FC Walters. Generic commutative separable algebras and cospans of graphs. *Theory and applications of categories*, 15(6):164–177, 2005.

[Zan15] Fabio Zanasi. *Interacting Hopf Algebras - The Theory of Linear Systems*. PhD thesis, École Normale Supérieure de Lyon, 2015.