Monoidal Width: Capturing Rank Width

Elena Di Lavore
Tallinn University of Technology

Paweł Sobociński
Tallinn University of Technology

Monoidal width was recently introduced by the authors as a measure of the complexity of decomposing morphisms in monoidal categories. We have shown that in a monoidal category of cospans of graphs, monoidal width and its variants can be used to capture tree width, path width and branch width. In this paper we study monoidal width in a category of matrices, and in an extension to a *different* monoidal category of open graphs, where the connectivity information is handled with matrix algebra and graphs are composed along edges instead of vertices. We show that here monoidal width captures rank width: a measure of graph complexity that has received much attention in recent years.

1 Introduction

Many applications of category theory rely on monoidal categories as algebras of processes [26, 15, 28, 18, 10, 25, 11, 17, 23, 27]. Morphisms are compound processes, defined as parallel and sequential compositions of simpler process components. The compositional nature of this modelling allows a compositional computation of the underlying semantics. But how efficient is this computation? Given two processes f and g, we can compute their semantics separately. However, computing the semantics of their sequential composition $f\,;g$ often requires an additional cost [36]. Indeed, the semantics of sequential composition often means resource sharing or synchronisation along the common boundary. This in turn carries a computational burden, dependent on the size of the boundary. On the other hand, computing the semantics of a parallel composition $f \otimes f'$ typically does not involve any resource sharing, as indicated by the wiring of the string diagrams, and thus typically does not require significant additional computational resources. Taking this into account, the choice of the *recipe* for a morphism in terms of parallel and sequential compositions influences the cost of computing its semantics. As shown in Figure 1, where vertical cuts represent sequential compositions and horizontal cuts represent parallel compositions, the same morphism can be defined in different ways with possibly different computational costs. Given a morphism, it is thefore desirable to find the least costly recipe of *decomposing* it in terms

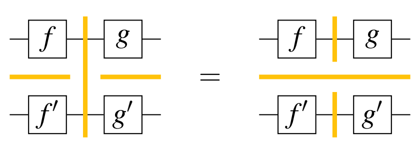


Figure 1: Two monoidal decompositions of the same morphism, the right one being the cheapest.

of more primitive components. We can rephrase the original question: what is the most efficient way to decompose a morphism in a monoidal category?

The authors recently proposed *monoidal width* [22] as a way of assigning a natural number to a morphism of a monoidal category, representing – roughly speaking – the cost of its most efficient decomposition. In turn, this is related to the cost of computing the semantics of this morphism.

Computing efficient decompositions is not a new problem. The graph theory literature abounds [6, 29, 38, 37, 39, 33, 20, 2, 3, 16] with notions of complexity of graphs that ultimately measure the difficulty

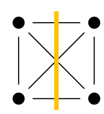
Preliminary Report. Final version to appear in:
ACT 2022

of decomposing a graph into smaller components by cutting along the vertices or the edges of the graph. Measures such as tree width [6, 29, 38], path width [37], branch width [39], clique width [20] and rank width [33] are motivated by algorithmic considerations. Probably the best known among several results that etablish links with algorithms [8, 9, 19], the following shows the importance of tree width.

Theorem (Courcelle [19]). *Every property expressible in the monadic second order logic of graphs can be tested in linear time on graphs with bounded tree width.*

The different notions of complexity for graphs vastly differ in low-level “implementation details” but they all share a similar underlying idea: that of defining decompositions and suitably measuring their efficiency. One of our contributions is to exhibit monoidal width as a unifying framework for graph measures based on a notion of decomposition. In fact, by choosing a suitable algebra of composition for graphs — i.e. choosing the right monoidal cateory — we recover some of these known measures as particular instances of monoidal width. We have previously captured [22] tree width, path width and branch width by instantiating monoidal width and two variants in a category of cospans of graphs.

In this paper we focus on rank width [33] – 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 its edges. The cost of a cut is the rank of the adjacency matrix that represents it, as shown in Figure 2. A useful intuition for rank width is that it is a kind of “Kolmogorov complexity” for graphs. For example, although the family of cliques has unbounded tree width, the connectivity of cliques is quite simple to describe: and, in fact, all cliques have rank width 1.



$$\mathsf{rank}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} = 1$$

Figure 2: A cut and its matrix in a rank decomposition.

To capture rank width as an instance of monoidal width, rather than taking cospans, we work in a different monoidal category of graphs. First introduced in [14], it was recently used [21] as a syntax for network games. This approach to computing with “open graphs” is more linear algebraic, building modularly on the theory of bialgebra, well known to be closely related to matrix algebra [41]. Indeed, the connectivity of graphs is handled with adjacency matrices, and boundary connections are matrices.

Related work. This manuscript, although self-contained, complements our previous work [22], where we considered tree width, path width and branch width as instances of monoidal width.

Previous syntactical approaches to graph widths are the work of Pudlák, Rödl and Savickỳ [35] and the work of Bauderon and Courcelle [5]. Their works consider different notions of graph decompositions, which lead to different notions of graph complexity. In particular, in [5], 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.

Recent abstract approaches focus on other graph widths. The work of Blume et. al. [7], characterises tree and path decompositions in terms of colimits. Abramsky et. al. [24] give a coalgebraic characterization of tree width of relational structures (and graphs in particular). Bumpus and Kocsis [13] also generalise tree width to the categorical setting, although their approach is far removed from ours.

Synopsis. Monoidal width is recalled in Section 2. In Section 3, we study the monoidal width of matrices. Section 4 recalls rank width and gives an equivalent recursive definition of it that will be useful as an intermediate step towards our main result, which is presented in Section 5.

Preliminaries. We use string diagrams [30, 40]: sequential and parallel compositions of f and g are drawn as in Figure 3, left and middle, respectively. Much of the bureaucracy, e.g. the interchange law $(f\,;g)\otimes(f'\,;g') = (f\otimes f')\,;(g\otimes g')$, disappears (Figure 3, right). *Props* [32, 31] are important examples of

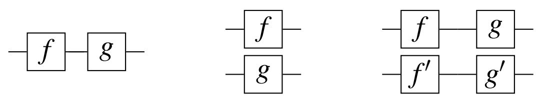


Figure 3: String diagrammatic notation.

monoidal categories. They are symmetric strict monoidal, with natural numbers as objects, and addition as monoidal product on objects. Roughly speaking, morphisms can be thought of as processes, and the objects (natural numbers) keep track of the number of inputs or outputs of a process.

2 Monoidal width

This section recalls the concept of monoidal width from [22]. Monoidal width records the cost of the most efficient way one can decompose a morphism into its atomic components, thus capturing—roughly speaking—its intrinsic structural complexity. A decomposition is a binary tree whose internal nodes are labelled with compositions or monoidal products, and whose leaves are labelled with atomic morphisms.

Definition 2.1 (Monoidal decomposition [22]). Let C be a monoidal category and $\mathscr{A}$ be a subset of its morphisms referred to as *atomic*. The set D_f of *monoidal decompositions* of $f\colon A\to B$ in C is defined:

$$\begin{array}{rcll} D_f & ::= & (f) & \text{if } f\in\mathscr{A} \\ & \mid & (d_1,\otimes,d_2) & \text{if } d_1\in D_{f_1},\, d_2\in D_{f_2} \text{ and } f = f_1\otimes f_2 \\ & \mid & (d_1,\,;_X,\,d_2) & \text{if } d_1\in D_{f_1\colon A\to X},\, d_2\in D_{f_2\colon X\to B} \text{ and } f = f_1\,;f_2 \end{array}$$

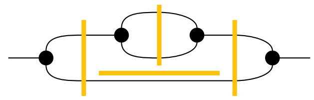
The cost of a decomposition depends on the operations and atoms present: each operation and each atomic morphism is associated with a cost, which we call weight. Roughly speaking, sequential composition is priced according to the size of the object the composition occurs over, while monoidal products are free. Finally, the weight of an atom is the application-specific cost of computing its semantics.

Definition 2.2 (Weight function [22]). Let C be a monoidal category and let $\mathscr{A}$ be a set of atoms for C. A weight function for $(\mathsf{C},\mathscr{A})$ is a function $\mathsf{w}\colon \mathscr{A}\cup\{\otimes\}\cup\mathsf{Obj}(\mathsf{C})\to\mathbb{N}$ such that

- $\mathsf{w}(X\otimes Y) = \mathsf{w}(X)+\mathsf{w}(Y)$,
- $\mathsf{w}(\otimes) = 0$.

Example 2.3. *Let* —●<$_1 : 1\to 2$ *and* >●—$_1 : 2\to 1$ *be the diagonal and codiagonal morphisms in a cartesian and cocartesian prop*[1] *s.t.* $\mathsf{w}($—●<$_1) = \mathsf{w}($>●—$_1) = 2$. *The following figure represents the monoidal decomposition of* —●< $;$ (—●< $\otimes$ —) $;$ (>●— $\otimes$ —) $;$ >●— *given by* (—●<, $;_2$, (((—●<, $;_2$, >●—), $\otimes$, —), $;_2$, >●—)).

[1] In a *cartesian prop* the $\otimes$ satisfies the universal property of products. Dually, in a *cocartesian prop*, the $\otimes$ satisfies the universal property of the coproduct.



The width of a decomposition is the cost of the most expensive node in the decomposition tree.

Definition 2.4 (Width of a monoidal decomposition [22])**.** Let w be a weight function for $(\mathsf{C},\mathscr{A})$. Let f be in C and $d \in D_f$. The width of d is defined recursively as follows:

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

As sketched in Example 2.3, decompositions can be seen as labelled trees (S,μ) where S is a tree and $\mu : \mathsf{vertices}(S) \to \mathscr{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{C})$ is a labelling function. With this we can restate the width as:

$$\mathsf{wd}(d) = \mathsf{wd}(S,\mu) := \max_{v \in \mathsf{vertices}(S)} \mathsf{w}(\mu(v))$$

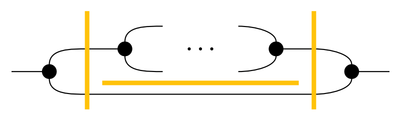
which may be familiar to those aquainted with graph widths.

Monoidal width is simply the width of the cheapest decomposition.

Definition 2.5 (Monoidal width [22])**.** Let w be a weight function for $(\mathsf{C},\mathscr{A})$ and f be in C. Then the *monoidal width* of f is $\mathsf{mwd}(f) := \min_{d \in D_f} \mathsf{wd}(d)$.

Example 2.6. *With the data of Example 2.3, define a family of morphisms $n \colon 1 \to 1$ inductively:*

- $1 := \text{—}_1$*;*
- $2 := -\!\!\bullet\!\!\!\subset \,;_2\, \supset\!\!\!\bullet\!\!-$*;*
- $n+1 := -\!\!\bullet\!\!\!\subset \,;_2\, (n \otimes 1) \,;_2\, \supset\!\!\!\bullet\!\!-$ *for* $n \geq 2$.



Each n has a monoidal decomposition of width n: the root node is the composition along the n wires in the middle. However, $\mathsf{mwd}(n) = 2$ *for any n, with an optimal decomposition shown above.*

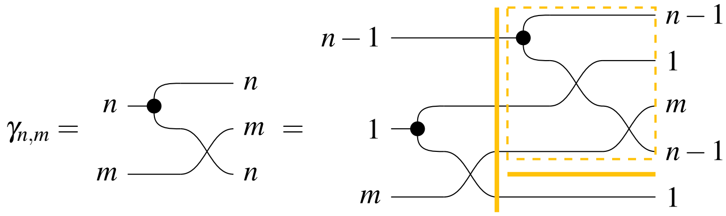
2.1 The width of copying

Before we begin with the original technical contributions of this paper in Section 3, we need to recall a technical result from [22] about decomposing copy morphisms. We consider symmetric monoidal categories equipped with such morphisms and show that copying n wires costs at most $n+1$.

Definition 2.7 (Copying)**.** Let X be a symmetric monoidal category with symmetries given by $\times_{X,Y}$. We say that X has *coherent copying* if there is a class of objects $\mathscr{C}_{\mathsf{X}} \subseteq \mathsf{Obj}(\mathsf{X})$, satisfying $X,Y \in \mathscr{C}_{\mathsf{X}}$ iff $X \otimes Y \in \mathscr{C}_{\mathsf{X}}$, such that every X in $\mathscr{C}_{\mathsf{X}}$ is endowed with a morphism $-\!\!\bullet\!\!\!\subset_X \colon X \to X \otimes X$. Moreover, $-\!\!\bullet\!\!\!\subset_{X\otimes Y} = (-\!\!\bullet\!\!\!\subset_X \otimes -\!\!\bullet\!\!\!\subset_Y) \,;\, (\text{—}_X \otimes \times_{X,Y} \otimes \text{—}_Y)$ for every $X,Y \in \mathscr{C}_{\mathsf{X}}$.

An example is any cartesian prop with $-\!\!\bullet\!\!\!\subset_n \colon n \to n+n$ given by the cartesian structure. We take $-\!\!\bullet\!\!\!\subset_X$, the symmetries $\times_{X,Y}$ and the identities —_X as atomic for all objects X,Y, i.e. the set of atomic morphisms is $\mathscr{A} = \{-\!\!\bullet\!\!\!\subset_X, \times_{X,Y}, \text{—}_X \;:\; X,Y \in \mathscr{C}_{\mathsf{X}}\}$. The weight function is $\mathsf{w}(-\!\!\bullet\!\!\!\subset_X) := 2 \cdot \mathsf{w}(X)$, $\mathsf{w}(\times_{X,Y}) := \mathsf{w}(X) + \mathsf{w}(Y)$ and $\mathsf{w}(\text{—}_X) := \mathsf{w}(X)$. In a prop, we take $\mathsf{w}(n) := n$. Note that $\mathsf{w}(-\!\!\bullet\!\!\!\subset_{X\otimes Y}) = 2 \cdot \mathsf{w}(X \otimes Y) = 2 \cdot (\mathsf{w}(X) + \mathsf{w}(Y))$, but utilising coherence we can do better, as illustrated below.

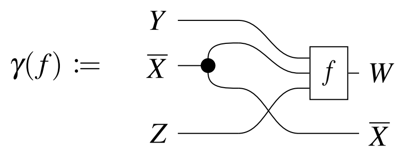
Example 2.8. *Let C be a prop with coherent copying and consider $-\!\!\bullet\!\!\!\subset_n \colon n \to 2n$. Let $\gamma_{n,m} := (-\!\!\bullet\!\!\!\subset_n \otimes \text{—}_m) \,;\, (\text{—}_n \otimes \times_{n,m}) \colon n+m \to n+m+n$. We can decompose $\gamma_{n,m}$ in terms of $\gamma_{n-1,m+1}$ (in the dashed box), $-\!\!\bullet\!\!\!\subset_1$ and $\times_{1,1}$ by cutting along at most $n+1+m$ wires:*



This allows us to decompose $-\!\!\bullet\!\!<_n = \gamma_{n,0}$ *cutting along at most* $n+1$ *wires. In particular,* $\mathsf{mwd}(-\!\!\bullet\!\!<_n) \leq n+1$.

The following result is a technical generalisation of the argument presented in Example 2.8.

Lemma 2.9 ([22])**.** *Let* X *be a symmetric monoidal category with coherent copying. Suppose that* $\mathscr{A}$ *contains* $-\!\!\bullet\!\!<_X$ *for* $X \in \mathscr{C}_{\mathsf{X}}$, *and* $\times_{X,Y}$ *and* $-_X$ *for* $X \in \mathsf{Obj}(\mathsf{X})$. *Let* $\overline{X} := X_1 \otimes \cdots \otimes X_n$, $f \colon Y \otimes \overline{X} \otimes Z \to W$ *and let* $d \in D_f$. *Let* $\gamma(f) := (-_Y \otimes -\!\!\bullet\!\!<_{\overline{X}} \otimes -_Z) \mathbin{;} (-_{Y \otimes \overline{X}} \otimes \times_{\overline{X},Z}) \mathbin{;} (f \otimes -_{\overline{X}})$.



There is a decomposition $\mathscr{C}(d)$ *of* $\gamma(f)$ *s.t.* $\mathsf{wd}(\mathscr{C}(d)) \leq \max\{\mathsf{wd}(d), \mathsf{w}(Y) + \mathsf{w}(Z) + (n+1) \cdot \max_{i=1,\dots,n} \mathsf{w}(X_i)\}$.

3 Monoidal width in matrices

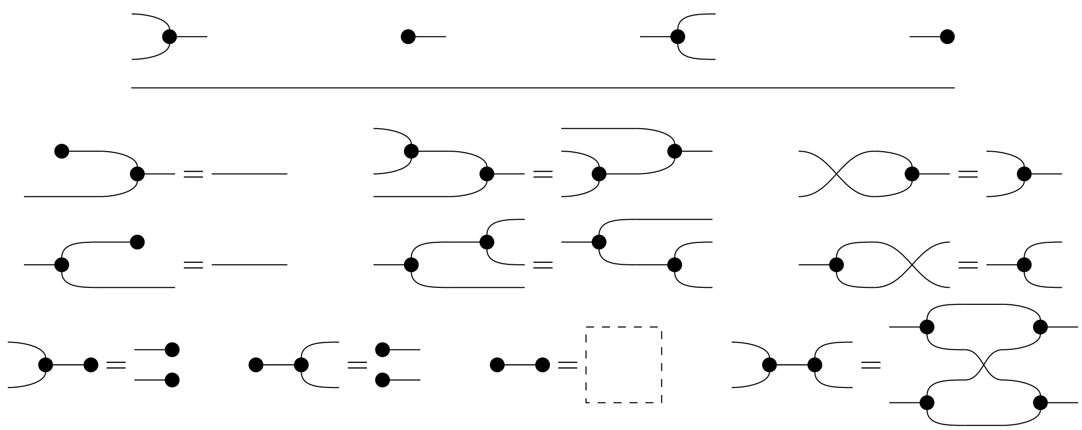


Figure 4: Bialgebra axioms

Given the ubiquity of matrix algebra, matrices are an obvious case study. Theorem 3.12 shows that the monoidal width of a matrix is, up to 1, the maximum of the ranks of its blocks.

Consider the monoidal category $\mathsf{Mat}_{\mathbb{N}}$ of matrices with entries in the natural numbers. The objects are natural numbers and morphisms from n to m are m by n matrices. 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. $A = \begin{pmatrix} A_1 & \mathbb{0} \\ \mathbb{0} & A_2 \end{pmatrix}$. On the other hand, a composition is related to the rank.

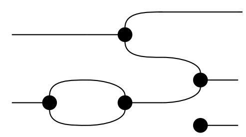
Lemma 3.1 ([34])**.** *Let* $A\colon n \to m$ *in* $\mathsf{Mat}_{\mathbb{N}}$. *Then* $\min\{k \in \mathbb{N} : A = B\,;_k C\} = \mathsf{rank}(A)$.

We first introduce a convenient syntax for matrices.

Proposition 3.2 ([41])**.** *The category* $\mathsf{Mat}_{\mathbb{N}}$ *is isomorphic to the prop* Bialg*, generated by* –●<: $1 \to 2$, –●: $1 \to 0$, >●–: $2 \to 1$ *and* ●–: $0 \to 1$ *and quotiented by bialgebra axioms (Figure 4).*

For the uninitiated reader, let us briefly explain this correspondence. Every morphism $f\colon n \to m$ in Bialg corresponds to a matrix $A = \mathsf{Mat}(f) \in \mathsf{Mat}_{\mathbb{N}}(m,n)$: 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.

Example 3.3. *The matrix* $\begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 0 & 0 \end{pmatrix} \in \mathsf{Mat}_{\mathbb{N}}(3,2)$ *corresponds to*



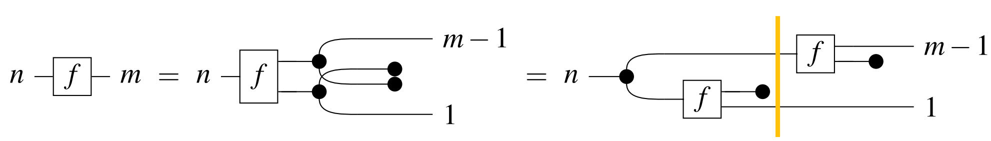
Definition 3.4. The atomic morphisms $\mathscr{A}$ are the generators of Bialg, with the symmetry and identity on 1: $\mathscr{A} = \{$–●<, –●, >●–, ●–, ×, $—_1\}$. The weight $\mathsf{w}\colon \mathscr{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{Bialg}) \to \mathbb{N}$ has $\mathsf{w}(n) := n$, for any $n \in \mathbb{N}$, and $\mathsf{w}(g) := \max\{m,n\}$, for $g\colon n \to m \in \mathscr{A}$.

3.1 Monoidal width in Bialg

The characterisation of the rank of a matrix in Lemma 3.1 hints at some relationship between the monoidal width of a matrix and its rank. In fact, we have Proposition 3.7, 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 3.5.

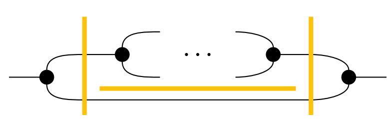
Proposition 3.5. *Let* P *be a cartesian and cocartesian prop. Suppose that* $—_1$, $–●<_1$, $>●–_1$, $–●_1$, $●–_1 \in \mathscr{A}$ *and* $\mathsf{w}(—_1) \leq 1$, $\mathsf{w}(–●<_1) \leq 2$, $\mathsf{w}(>●–_1) \leq 2$, $\mathsf{w}(–●_1) \leq 1$ *and* $\mathsf{w}(●–_1) \leq 1$. *Suppose that, for every* $g\colon 1 \to 1$, $\mathsf{mwd}(g) \leq 2$. *Let* $f\colon n \to m$ *be a morphism in* P. *Then* $\mathsf{mwd}(f) \leq \min\{m,n\} + 1$.

Proof sketch. The proof proceeds by induction on $\max\{m,n\}$. The base cases are easily checked. The inductive step relies on the fact that, applying Lemma 2.9, if $n < m$, we can decompose f as shown below by cutting at most $n+1$ wires or, if $m < n$, in the symmetric way by cutting at most $m+1$ wires.



The details can be found in Appendix A. □

We can apply the former result to Bialg and obtain Proposition 3.7 because the width of 1×1 matrices, which are numbers, is at most 2. This follows from the reasoning in Example 2.6 as we can write every natural number $k\colon 1 \to 1$ as the following composition:



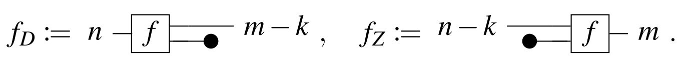
Lemma 3.6. *Let* $k\colon 1 \to 1$ *in* Bialg. *Then,* $\mathsf{mwd}(k) \leq 2$.

Proposition 3.7. *Let $f\colon n \to m$ in* Bialg*. Then,* $\mathsf{mwd} f \leq \mathsf{rank}(\mathsf{Mat} f) + 1$*. 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* $\mathsf{rank}(\mathsf{Mat} f) \leq \mathsf{mwd} f$*.*

Proof sketch. This result follows from Lemma 3.1 and Proposition 3.5, which we can apply thanks to Lemma 3.6. The details are in Appendix A. □

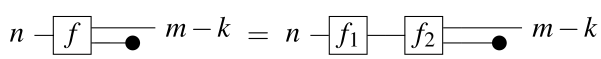
The bounds given by Proposition 3.7 can be improven when we have a $\otimes$-decomposition of a matrix, i.e. we can write $f = f_1 \otimes \ldots \otimes f_k$, to obtain Proposition 3.9. The latter relies on Lemma 3.8, which shows that discarding inputs or outputs cannot increase the monoidal width of a morphism in Bialg.

Lemma 3.8. *Let $f\colon n \to m$ in* Bialg *and $d \in D_f$. Let $f_D := f\,;(\text{—}_{m-k} \otimes \text{-}\!\bullet_k)$ and $f_Z := (\text{—}_{n-k'} \otimes \bullet\!\text{-}_{k'})\,; f$, where $\text{-}\!\bullet_k\colon k \to 0$ is the discard morphism with $k \leq m$ and $\bullet\!\text{-}_{k'}\colon 0 \to k$ is the zero morphism with $k' \leq n$.*



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

Proof sketch. By induction. The base cases are easy. If $f = f_1\,; f_2$, use the inductive hypothesis on f_2.



The $f = f_1 \otimes f_2$ case is similar. The details are in Appendix A. □

Proposition 3.9. *Let $f\colon n \to m$ in* Bialg *and $d' = (d'_1, ;_k, d'_2) \in D_f$. Suppose there are f_1 and f_2 such that $f = f_1 \otimes f_2$. Then, there is $d = (d_1, \otimes, d_2) \in D_f$ such that* $\mathsf{wd}(d) \leq \mathsf{wd}(d')$*.*

Proof sketch. By Lemma 3.1, $\mathsf{rank}(\mathsf{Mat} f_1) + \mathsf{rank}(\mathsf{Mat} f_2) = \mathsf{rank}(\mathsf{Mat}(f_1 \otimes f_2)) \leq k$ and, by Proposition 3.7, there is a monoidal decomposition d_i of f_i such that $\mathsf{wd}(d_i) \leq \mathsf{rank}(\mathsf{Mat} f_i) + 1$. Then, $\mathsf{wd}(d) := \mathsf{wd}((d_1, \otimes, d_2)) \leq \max\{\mathsf{rank}(\mathsf{Mat} f_1), \mathsf{rank}(\mathsf{Mat} f_2)\} + 1 \leq \mathsf{rank}(\mathsf{Mat} f_1) + \mathsf{rank}(\mathsf{Mat} f_2)$ whenever $\mathsf{rank}(\mathsf{Mat} f_1), \mathsf{rank}(\mathsf{Mat} f_2) > 0$. We apply Lemma 3.8 to obtain the same result if $\mathsf{rank}(\mathsf{Mat} f_1) = 0$ or $\mathsf{rank}(\mathsf{Mat} f_2) = 0$. The details are in Appendix A. □

We summarise Proposition 3.9 and Proposition 3.7 in Corollary 3.10.

Corollary 3.10. *Let $f = f_1 \otimes \ldots \otimes f_k$ in* Bialg*. Then,* $\mathsf{mwd}(f) \leq \max_{i=1,\ldots,k} \mathsf{rank}(\mathsf{Mat}(f_i)) + 1$*. Moreover, if f_i are not $\otimes$-decomposable, then* $\max_{i=1,\ldots,k} \mathsf{rank}(\mathsf{Mat}(f_i)) \leq \mathsf{mwd} f$*.*

Proof. By Proposition 3.9 there is a decomposition of f of the form $d = (d_1, \otimes, \cdots (d_{k-1}, \otimes, d_k))$, where we can choose d_i to be a minimal decomposition of f_i. Then, $\mathsf{mwd}(f) \leq \mathsf{wd}(d) = \max_{i=1,\ldots,k} \mathsf{wd}(d_i)$. By Proposition 3.7, $\mathsf{wd}(d_i) \leq r_i + 1$. Then, $\mathsf{mwd}(f) \leq \max\{r_1, \ldots, r_k\} + 1$. Moreover, if f_i are not $\otimes$-decomposable, Proposition 3.7 gives also a lower bound on their monoidal width: $\mathsf{rank}(\mathsf{Mat}(f_i)) \leq \mathsf{mwd} f_i$; and we obtain that $\max_{i=1,\ldots,k} \mathsf{rank}(\mathsf{Mat}(f_i)) \leq \mathsf{mwd} f$. □

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 3.11 shows that every morphism in Bialg has a unique $\otimes$-decomposition.

Proposition 3.11. *Let* C *be a monoidal category whose monoidal unit* 0 *is both initial and terminal, and whose objects are a unique factorization monoid. Let f be a morphism in* C*. Then f has a unique $\otimes$-decomposition.*

Proof. See Appendix A for a proof sketch and the proof. □

Our main result in this section follows from Corollary 3.10 and Proposition 3.11, which can be applied to Bialg because 0 is both terminal and initial, and the objects, being a free monoid, are a unique factorization monoid.

Theorem 3.12. *Let* $f = f_1 \otimes \ldots \otimes f_k$ *be a morphism in* Bialg *and its unique* $\otimes$*-decomposition given by Proposition 3.11, with* $r_i = \mathsf{rank}(\mathsf{Mat}(f_i))$. *Then* $\max\{r_1, \ldots, r_k\} \leq \mathsf{mwd}(f) \leq \max\{r_1, \ldots, r_k\} + 1$.

Proof. This result is obtained by applying Corollary 3.10 to the $\otimes$-decomposition given by Proposition 3.11, which can be applied because, in Bialg, 0 is both terminal and initial, and the objects, being a free monoid, are a unique factorization monoid. □

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.

4 Graphs and rank width

Here we recall rank width [33] for undirected graphs.

Definition 4.1. An *undirected graph* $G = (V, E, \mathsf{ends})$ is given by a set of edges E, a set of vertices V and a function ends: $E \to \wp_{\leq 2}(V)$ that gives the endpoints of each edge. We consider graphs *up to isomorphism*, or *abstract graphs*, thus the set of vertices can be fully characterised by its cardinality. An abstract graph can be equivalently given by an adjacency matrix $[G]$, where $G \in \mathsf{Mat}_{\mathbb{N}}(n,n)$ and n is the number of vertices. The equivalence class of adjacency matrices is defined by the equivalence relation

$$G \sim H \quad \text{iff} \quad G + G^\top = H + H^\top.$$

We will refer to abstract undirected graphs as simply graphs.

Definition 4.2. A *path* in a graph G is a sequence of edges $(e_1, \ldots, e_k)$ together with a sequence of distinct vertices $(v_1, \ldots, v_{k+1})$ of G such that, for every $i = 1, \ldots, k$, $\mathsf{ends}(e_i) = \{v_i, v_{i+1}\}$. A *tree* is a graph such that there is a unique path between any two of its vertices. Two vertices v and w in a graph G are *neighbours* if G has an edge between them. The *leaves* of a tree are those vertices with at most one neighbour. A *subcubic tree* is a tree where each vertex has between one and three neighbours.

A rank decomposition for a graph G is a tree whose leaves are labelled with the vertices of G.

Definition 4.3 ([33])**.** 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 two 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 4.4 (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: $\mathsf{ord}(b) := \mathsf{rank}(X_b)$.

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

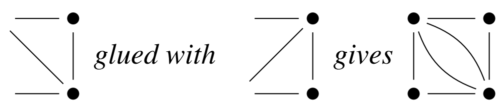
$$\mathsf{rwd}(G) := \min_{(Y,r)} \mathsf{wd}(Y,r).$$

4.1 Graphs with dangling edges

As intermediate step between rank decompositions and monoidal decompositions, we introduce recursive rank decompositions of *graphs with dangling edges* and we prove that they give a notion of width that is equivalent to rank width. Similar recursive characterisations were done for tree decompositions in [4] and for path and branch decompositions in [22]. We first need a notion of graph that is equipped with some "open" edges along which it can be glued with other graphs.

Definition 4.6. 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 $B = \mathsf{boundary}(\Gamma)$.

Example 4.7. *Two graphs with the same ports, as illustrated below, can be "glued" together:*



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: $T_\Gamma \ ::= \ (\Gamma') \ \mid \ (T_\Gamma, \Gamma', T_\Gamma)$. Given $T \in T_\Gamma$, the label function λ takes a decomposition and returns the graph with dangling edges at the root: $\lambda(T_1, \Gamma, T_2) := \Gamma$ and $\lambda(\Gamma) := \Gamma$.

Definition 4.8 (Recursive rank decomposition)**.** Let $\Gamma = ([G], B)$ be a graph with dangling edges, where $G \in \mathsf{Mat}_{\mathbb{N}}(k,k)$ and $B \in \mathsf{Mat}_{\mathbb{N}}(k,n)$. A recursive rank decomposition of Γ is $T \in T_\Gamma$ where either: Γ has at most one vertex and $T = (\Gamma)$; or $T = (T_1, \Gamma, T_2)$ and $T_i \in T_{\Gamma_i}$ are recursive 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 $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$.

As with before, the *recursive* rank width of a graph is the width of its cheapest decomposition.

Definition 4.9. Let T be a recursive rank decomposition of $\Gamma = ([G], B)$. Define the width of T recursively: if $T = (\Gamma)$, $\mathsf{wd}(T) := \mathsf{rank}(B)$, and, if $T = (T_1, \Gamma, T_2)$, $\mathsf{wd}(T) := \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(B)\}$ Expanding this expression, we obtain $\mathsf{wd}(T) = \max_{T' \text{ subtree of } T} \mathsf{rank}(\mathsf{boundary}(\lambda(T')))$. The *recursive rank width* of Γ is defined by the min-max formula $\mathsf{rrwd}(\Gamma) := \min_T \mathsf{wd}(T)$.

We show that recursive rank width is the same as rank width, up to the rank of the boundary of the graph.

Proposition 4.10. *Let $\Gamma = ([G], B)$ be a graph with dangling edges and (Y,r) be a rank decomposition of G. Then, there is a recursive rank decomposition $\mathscr{I}(Y,r)$ of Γ s.t.* $\mathsf{wd}(\mathscr{I}(Y,r)) \leq \mathsf{wd}(Y,r) + \mathsf{rank}(B)$.

Proof. See Appendix B. □

Before proving the lower bound for recursive rank width, we need a technical lemma that relates the width of a graph with that of its subgraphs.

Lemma 4.11. *Let T be a recursive rank decomposition of $\Gamma = ([G], B)$. Let T' be a 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,* $\mathsf{rank}(B') = \mathsf{rank}(A' \mid C_L^\top \mid C_R)$.

Proof. See Appendix B. □

Proposition 4.12. *Let T be a recursive 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 $\mathscr{I}^\dagger(T)$ of G such that $\mathsf{wd}(\mathscr{I}^\dagger(T)) \leq \mathsf{wd}(T)$.*

Proof. See Appendix B. □

From Proposition 4.12 and Proposition 4.10 we conclude the following result.

Theorem 4.13. *Let $\Gamma = ([G], B)$. Then,* $\mathsf{rwd}(G) \leq \mathsf{rrwd}(\Gamma) \leq \mathsf{rwd}(G) + \mathsf{rank}(B)$.

5 Monoidal width and rank width

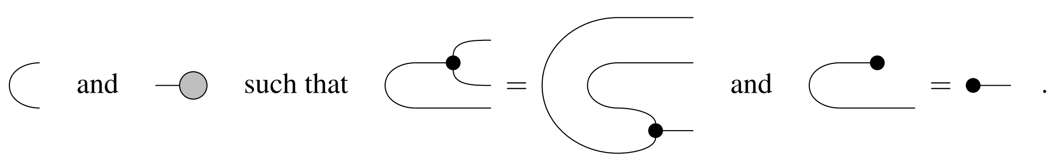
This section contains our main results. We prove that monoidal width in the prop of graphs Grph [14] corresponds to rank width, up to a constant multiplicative factor of 2.

We start by introducing the algebra of graphs with boundaries and its diagrammatic syntax [21]. A graph with boundaries is a graph together with two matrices L and R that record the connectivity of the vertices with the left and right boundary, a matrix P that records the passing wires from the left boundary to the right one and a matrix F that records the wires from the right boundary to itself.

Definition 5.1 ([21])**.** A *graph with boundaries* $g\colon n \to m$ is defined as $g = ([G], L, R, P, [F])$, where $[G]$ is the adjacency matrix of a graph on k vertices, with $G \in \mathsf{Mat}_{\mathbb{N}}(k,k)$; $L \in \mathsf{Mat}_{\mathbb{N}}(k,n)$, $R \in \mathsf{Mat}_{\mathbb{N}}(k,m)$, $P \in \mathsf{Mat}_{\mathbb{N}}(m,n)$ and $F \in \mathsf{Mat}_{\mathbb{N}}(m,m)$ recording connectivity information as explained above. 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 [21], forming a symmetric monoidal category BGraph. 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 4).

Definition 5.2 ([14])**.** The prop of graphs Grph is obtained by adding to Bialg the generators $\cup\colon 0 \to 2$ and $v\colon 1 \to 0$ with the equations below.



These equations mean, in particular, that the cup transposes matrices (Figure 5, left) and that we can express the equivalence relation of adjacency matrices: $G \sim H$ iff $G + G^\top = H + H^\top$ (Figure 5, right).

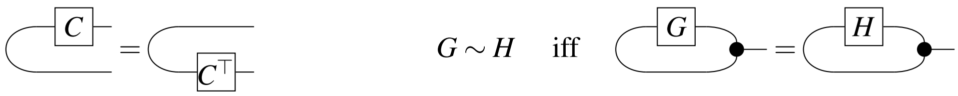
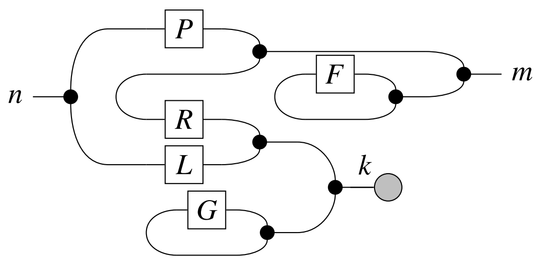


Figure 5: Adding the cup.

Proposition 5.3 ([21], Theorem 23)**.** *The prop of graphs* Grph *is isomorphic to the prop* BGraph.

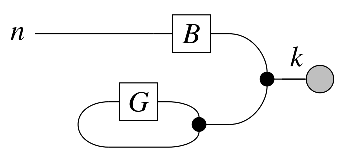
Proposition 5.3 means that the morphisms in Grph can be written in the following normal form



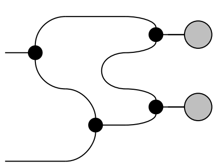
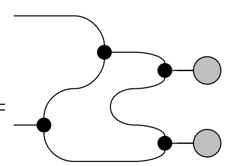
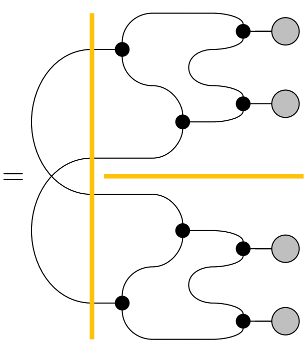
The prop Grph is more expressive than graphs with dangling edges (Definition 4.6): its morphisms can have edges between the boundaries as well. In fact, graphs with dangling edges can be seen as morphisms $n \to 0$ in Grph.

Example 5.4. *A graph with dangling edges* $\Gamma = ([G], B)$ *can be represented as a morphism in* Grph

$$g = ([G], B, ¡, !, [()]) =$$



We can now formalise the intuition of glueing graphs with dangling edges as explained in Example 4.7. 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.*

$g_1 =$  $g_2 =$  $\cup_2 ; (g_1 \otimes g_2) =$ 

5.1 Rank width in open graphs

The technical content of our main result (Theorem 5.12) is split in two: an upper and a lower bound.

As in the prop of matrices Bialg, the cost of composing along n wires is n. All morphisms in Grph are chosen as atomic. One could restrict this to those with at most one vertex without affecting the results.

Definition 5.5. Let the set of *atomic morphisms* $\mathscr{A}$ be the set of all the morphisms of Grph. The *weight function* $\mathsf{w} \colon \mathscr{A} \cup \{\otimes\} \cup \mathsf{Obj}(\mathsf{Grph}) \to \mathbb{N}$ is defined, on objects n, as $\mathsf{w}(n) := n$; and, on morphisms $g \in \mathscr{A}$, as $\mathsf{w}(g) := k$, where k is the number of vertices of g.

Note that the monoidal width of g is bounded by the number of its vertices.

The upper bound (Proposition 5.8) is established by associating to each recursive rank decomposition a suitable monoidal decomposition. This mapping is defined inductively, given the inductive nature of both these structures. Given a recursive rank decomposition of a graph Γ, we can construct a decomposition of its corresponding morphism g as shown by the first equality in Figure 6. However, this

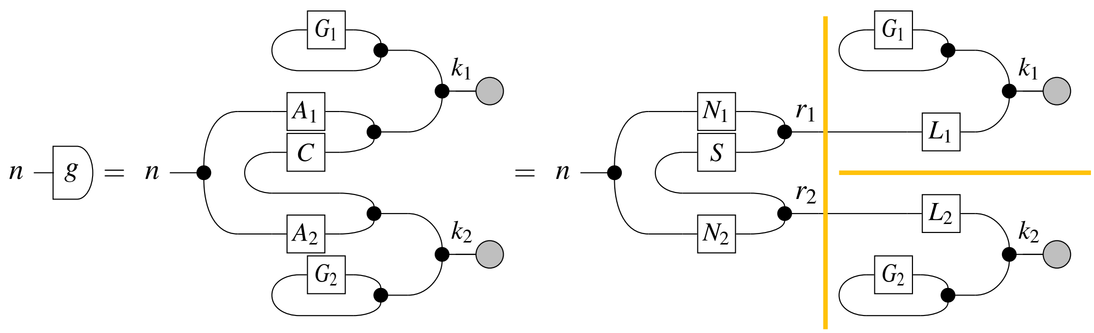
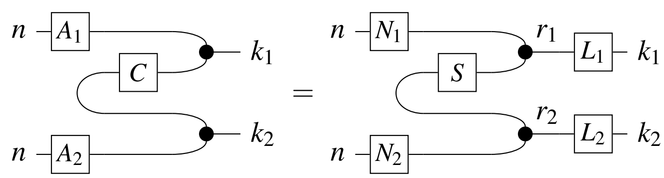


Figure 6: First step of a monoidal decomposition given by a recursive rank decomposition

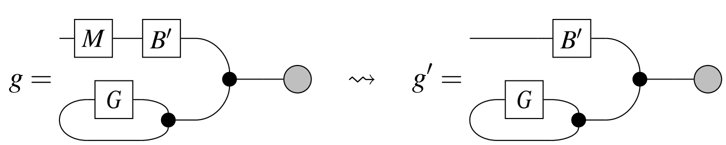
decomposition is not optimal as it cuts along the number of vertices $k_1 + k_2$. But we can do better thanks to Lemma 5.6, which shows that we can cut along the ranks, $r_1 = \mathsf{rank}(A_1 \mid C)$ and $r_2 = \mathsf{rank}(A_2 \mid C^\top)$, of the boundaries of the induced subgraphs to obtain the second equality in Figure 6.



Lemma 5.6. *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 $(A_2 \mid C^\top) = L_2 \cdot (N_2 \mid S^\top \cdot L_1^\top)$.*

Proof. See Appendix C. □

Once we have performed the cuts in Figure 6 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 recursive rank decompositions of the old subgraphs into decompositions of the new ones, as shown in Lemma 5.7. More explicitly, when M has full rank, if we have a recursive rank decomposition of $\Gamma = ([G], B' \cdot M)$, which corresponds to g below left, we can obtain one of $\Gamma' = ([G], B')$, which corresponds to g' below right, of the same width.



Lemma 5.7. *Let T be a recursive rank decomposition of $\Gamma = ([G], B)$ and $B = B' \cdot M$, with M that has full rank. Then, there is a recursive rank decomposition T' of $\Gamma' = ([G], B')$ such that $\mathsf{wd}(T) = \mathsf{wd}(T')$ and such that T and T' have the same underlying tree structure.*

Proof. See Appendix C. □

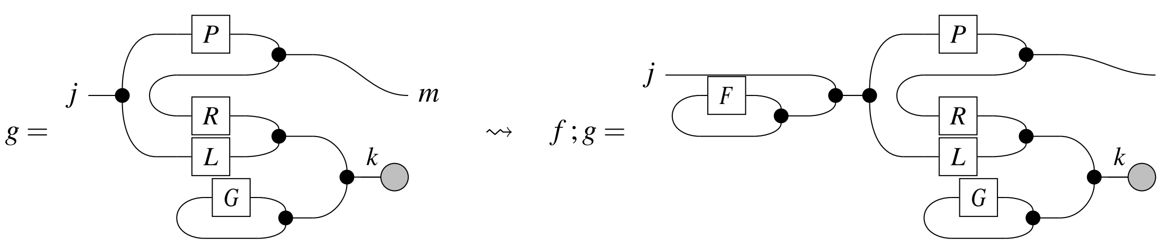
With the above ingredients, we can show that rank width bounds monoidal width from above.

Proposition 5.8. *Let $\Gamma = ([G], B)$ be a graph with dangling edges and $g \colon n \to 0$ be the morphism in* Grph *corresponding to* Γ. *Let T be a recursive rank decomposition of* Γ. *Then, there is a monoidal decomposition $\mathscr{R}^\dagger(T)$ of g such that* $\mathsf{wd}(\mathscr{R}^\dagger(T)) \leq 2 \cdot \mathsf{wd}(T)$.

Proof sketch. The proof proceeds by induction on T. The base cases are easily checked and the inductive step relies on the decomposition of g in Figure 6, which we can write thanks to Lemma 5.6. Applying the inductive hypothesis and Lemma 5.7, the width of this decomposition can be bounded by $\max\{r_1 + r_2, 2 \cdot \mathsf{wd}(T_1), 2 \cdot \mathsf{wd}(T_2)\} \leq 2 \cdot \mathsf{wd}(T)$, where $T = (T_1, \Gamma, T_2)$. The details are in Appendix C. □

Proving the lower bound is similarly involved and follows a similar proof structure. From a monoidal decomposition we construct inductively a recursive 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 recursive rank decompositions of the induced subgraphs into ones of two subgraphs that satisfy the conditions of Definition 4.8.

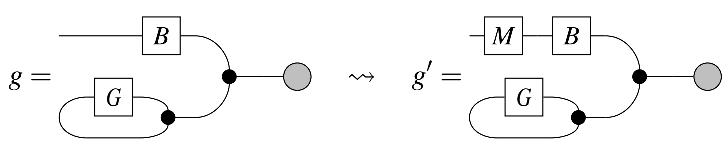
Applying the inductive hypothesis gives us a recursive rank decomposition of $\Gamma = ([G], (L \mid R))$, which is associated to g below left, and we need to construct one of $\Gamma' := ([G + L \cdot F \cdot L^\top], (L \mid R + L \cdot (F + F^\top) \cdot P^\top))$, which is associated to $f \,;g$ below right, of at most the same width.



Lemma 5.9. *Let T be a recursive 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 a recursive rank decomposition T' of Γ' such that* $\mathsf{wd}(T') \leq \mathsf{wd}(T)$.

Proof. See Appendix C. □

In order to obtain the subgraphs of the desired shape we need to add some extra connections to the boundaries. We have a recursive rank decomposition of $\Gamma = ([G], B)$, which corresponds to g below left, and we need one of $\Gamma' = ([G], B \cdot M)$, which corresponds to g' below right, of at most the same width.



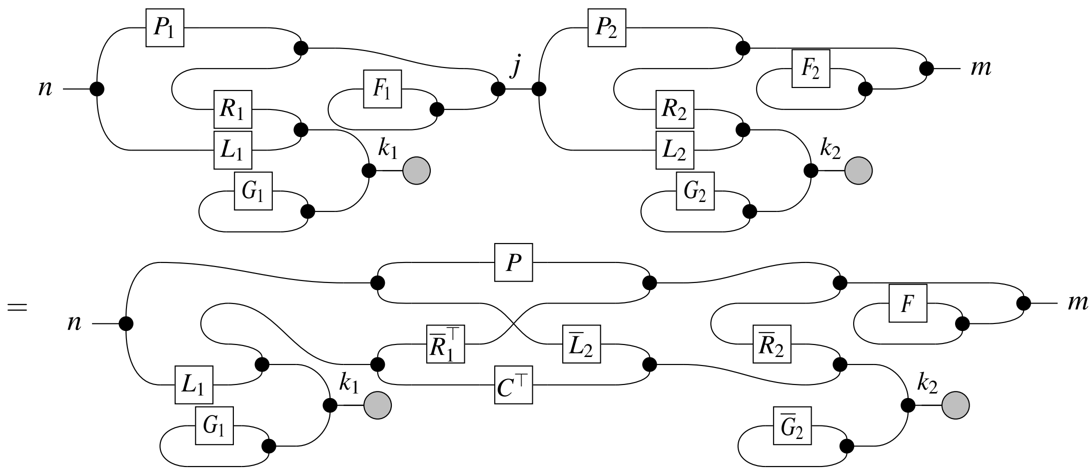
The following result and its proof are very similar to Lemma 5.7.

Lemma 5.10. *Let T be a recursive rank decomposition of $\Gamma = ([G], B)$ and let $B' = B \cdot M$. Then, there is a recursive rank decomposition T' of $\Gamma' = ([G], B')$ such that $\mathsf{wd}(T') \leq \mathsf{wd}(T)$ and such that T and T' have the same underlying tree structure. Moreover, if M has full rank, then $\mathsf{wd}(T') = \mathsf{wd}(T)$.*

Proof. See Appendix C. □

Proposition 5.11. *Let* $g = ([G], L, R, P, [F])$ *in* Grph *and* $d \in D_g$. *Let* $\Gamma = ([G], (L \mid R))$. *Then, there is a recursive rank decomposition* $\mathscr{R}(d)$ *of* Γ *s.t.* $\mathsf{wd}(\mathscr{R}(d)) \leq 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rank}(L), \mathsf{rank}(R)\}$.

Proof sketch. The proof proceeds by induction on d. The base case is easily checked, while the inductive steps are a bit more involved. If $d = (d_1, ;_j, d_2)$, then there are $g_i = ([G_i], L_i, R_i, P_i, [F_i])$ such that $g = g_1 ; g_2$ and we can write g as follows.



In order to build a recursive rank decomposition of Γ, we need recursive rank decompositions of $\overline{\Gamma}_i = ([\overline{G}_i], \overline{B}_i)$, but we can obtain recursive rank decompositions of $\Gamma_i = ([G_i], (L_i \mid R_i))$ by applying only induction. Thanks to Lemma 5.9, we obtain a recursive rank decomposition of $\Gamma_2' = ([G_2 + L_2 \cdot F_1 \cdot L_2^\top], (L_2 \mid R_2 + L_2 \cdot (F_1 + F_1^\top) \cdot P_2^\top))$. Lastly, we apply Lemma 5.10 to get recursive rank decompositions T_i of $\overline{\Gamma}_i$. Thanks to these, we can bound the width of $T := (T_1, \Gamma, T_2)$:

$$\mathsf{wd}(T) \leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2), j, \mathsf{rank}(L), \mathsf{rank}(R)\} =: 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rank}(L), \mathsf{rank}(R)\}.$$

If $d = (d_1, \otimes, d_2)$, we proceed similarly. For the details, see Appendix C. □

From Proposition 5.8, Proposition 5.11 and Theorem 4.13, we obtain our main result.

Theorem 5.12. *Let* G *be a graph and let* $g = ([G], ¡, ¡, (\,), [(\,)])$ *be the corresponding morphism of* Grph. *Then,* $\frac{1}{2} \cdot \mathsf{rwd}(G) \leq \mathsf{mwd}(g) \leq 2 \cdot \mathsf{rwd}(G)$.

6 Conclusions and future work

We have shown that monoidal width, in a suitable category of graphs composable along “open” edges, yields rank width; a well-known measure from the graph theory literature.

Our goal with this line of research is to develop a generic, abstract “decomposition theory”. We will study other graph widths like clique width [20] and twin width [12], as well as go beyond graphs: e.g. by focussing on tree width for hypergraphs and relational structures [1], branch width for matroids and widths for directed graphs. A part of “decomposition theory” means going beyond width as a mere number – in fact we believe that in each case the identification of a suitable monoidal category as an *algebra* of open graph structures is itself a worthwhile contribution. Indeed, having such an algebra means that a decomposition, rather than an ad hoc concept-specific construction, becomes more of a mathematical object in its own right. Such compositional algebras will add to the quiver of compositional structures of applied category theory; for example serving as syntax for more sophisticated applications [21].

Acknowledgements. Elena Di Lavore and Paweł Sobociński were supported by the European Union through the ESF funded Estonian IT Academy research measure (2014-2020.4.05.19-0001). This work was also supported by the Estonian Research Council grant PRG1210.

References

[1] 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.

[2] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[3] Dorit Aharonov, Zeph Landau, and Johann Makowsky. The quantum fft can be classically simulated, 2006. `arXiv:quant-ph/0611156`.

[4] Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *Journal of the ACM (JACM)*, 40(5):1134–1164, 1993.

[5] Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20(1):83–127, 1987.

[6] Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973.

[7] Christoph Blume, HJ Sander Bruggink, Martin Friedrich, and Barbara König. Treewidth, pathwidth and cospan decompositions. *Electronic Communications of the EASST*, 41, 2011.

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

[9] Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[10] Guillaume Boisseau and Paweł Sobociński. String diagrammatic electrical circuit theory. *CoRR*, abs/2106.07763, 2021. URL: `https://arxiv.org/abs/2106.07763`, `arXiv:2106.07763`.

[11] 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`.

[12] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width i: tractable fo model checking. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 601–612. IEEE, 2020.

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

[14] Apiwat Chantawibul and Paweł Sobociński. Towards compositional graph theory. *Electronic Notes in Theoretical Computer Science*, 319:121–136, 2015.

[15] Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. `arXiv:1709.00322`, `doi:10.1017/S0960129518000488`.

[16] Maria Chudnovsky and Paul Seymour. A well-quasi-order for tournaments. *Journal of Combinatorial Theory, Series B*, 101(1):47–53, 2011.

[17] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes - A first course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[18] Cole Comfort and Aleks Kissinger. A graphical calculus for lagrangian relations. *CoRR*, abs/2105.06244, 2021. URL: `https://arxiv.org/abs/2105.06244`, `arXiv:2105.06244`.

[19] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.

[20] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[21] Elena Di Lavore, Jules Hedges, and Paweł Sobociński. Compositional modelling of network games. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[22] Elena Di Lavore and Paweł Sobociński. Monoidal Width: Unifying Tree Width, Path Width and Branch Width, 2022. arXiv:2202.07582.

[23] 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.

[24] Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

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

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

[27] 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.

[28] 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.

[29] Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8(1-2):171–186, 1976.

[30] André Joyal and Ross Street. The geometry of tensor calculus, i. *Advances in mathematics*, 88(1):55–112, 1991.

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

[32] Saunders Mac Lane. Categorical algebra. *Bull. Amer. Math. Soc.*, 71:40–106, 1965.

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

[34] R Piziak and PL Odell. Full rank factorization of matrices. *Mathematics magazine*, 72(3):193–201, 1999.

[35] Pavel Pudlák, Vojtěch Rödl, and Petr Savickỳ. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988.

[36] Julian Rathke, Paweł Sobociński, and Owen Stephens. Compositional reachability in petri nets. In *International Workshop on Reachability Problems*, pages 230–243. Springer, 2014.

[37] Neil Robertson and Paul D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[38] Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

[39] Neil Robertson and Paul D. Seymour. Graph minors. X. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

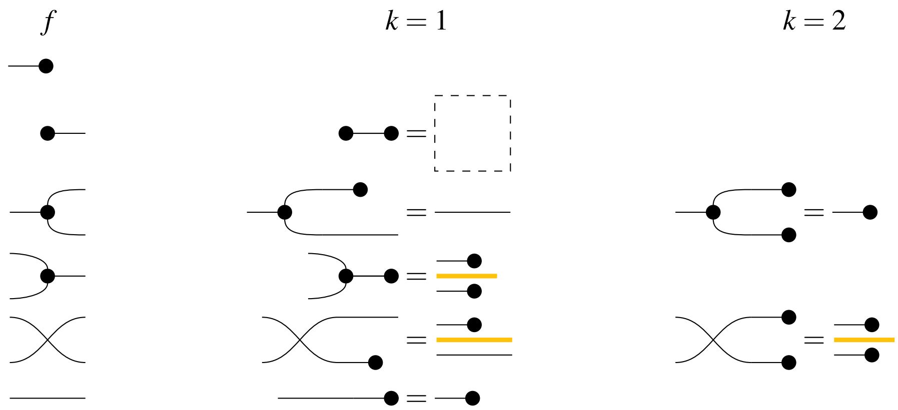
[40] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

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

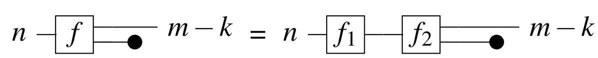
A Matrices

Proof of Lemma 3.8. By induction on d.

If the decomposition has only one node, $d = (f)$, then f is one of the six atoms in the first column of the table below. The second and third columns show the decompositions of f_D for $k = 1$ and $k = 2$.

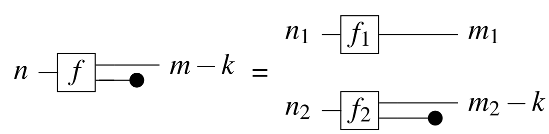


If the decomposition starts with a composition node, $d = (d_1, ;, d_2)$, then $f = f_1 \,; f_2$, with d_i monoidal decomposition of f_i.

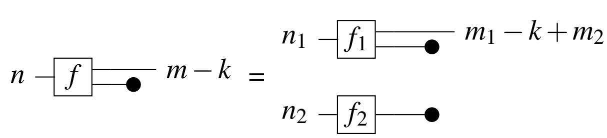


By induction, there is a monoidal decomposition $\mathscr{D}(d_2)$ of $f_2 \,; (-_{m-k} \otimes {-\bullet}_k)$ such that $\mathsf{wd}(\mathscr{D}(d_2)) \leq \mathsf{wd}(d_2)$. Let $\mathscr{D}(d) := (d_1, ;, \mathscr{D}(d_2))$. Then, $\mathscr{D}(d)$ is a monoidal decomposition of $f \,; (-_{m-k} \otimes {-\bullet}_k)$ because $f \,; (-_{m-k} \otimes {-\bullet}_k) = f_1 \,; (f_2 \,; -_{m-k} \otimes {-\bullet}_k)$.

If the decomposition starts with a tensor node, $d = (d_1, \otimes, d_2)$, 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 $f \,; (-_{m-k} \otimes {-\bullet}_k) = f_1 \otimes (f_2 \,; (-_{m_2-k} \otimes {-\bullet}_k))$.



By induction, there is a monoidal decomposition $\mathscr{D}(d_2)$ of $f_2 \,; (-_{m-k} \otimes {-\bullet}_k)$ such that $\mathsf{wd}(\mathscr{D}(d_2)) \leq \mathsf{wd}(d_2)$. Then, $\mathscr{D}(d) := (d_1, \otimes, \mathscr{D}(d_2))$ is a monoidal decomposition of $f \,; (-_{m-k} \otimes {-\bullet}_k)$. If $k > m_2$, then $f \,; (-_{m-k} \otimes {-\bullet}_k) = (f_1 \,; (-_{m_1-k+m_2} \otimes {-\bullet}_{k-m_2})) \otimes (f_2 \,; {-\bullet}_{m_2})$.



By induction, there are monoidal decompositions $\mathscr{D}(d_i)$ of $f_1 \,; (-_{m_1-k+m_2} \otimes {-\bullet}_{k-m_2})$ and $f_2 \,; {-\bullet}_{m_2}$ such that $\mathsf{wd}(\mathscr{D}(d_i)) \leq \mathsf{wd}(d_i)$. Then, $\mathscr{D}(d) := (\mathscr{D}(d_1), \otimes, \mathscr{D}(d_2))$ is a monoidal decomposition of $f \,; (-_{m-k} \otimes {-\bullet}_k)$.

The second inequality is proven using the same inductive argument. $\square$

Proof of Proposition 3.9. By hypothesis, d' is a monoidal decomposition of f. Then, there are g and h such that $f_1 \otimes f_2 = f = g\,;h$. By Proposition 3.7, there are monoidal decompositions d_i of f_i with $\mathsf{wd}(d_i) \leq r_i + 1$, where $r_i := \mathsf{rank}(\mathsf{Mat} f_i)$. By properties of the rank, $r_1 + r_2 = \mathsf{rank}(\mathsf{Mat} f)$. By Lemma 3.1, $\mathsf{rank}(\mathsf{Mat} f) \leq k$. There are two cases.

If $r_i > 0$, then $r_1 + r_2 \geq \max\{r_1, r_2\} + 1$. Then,

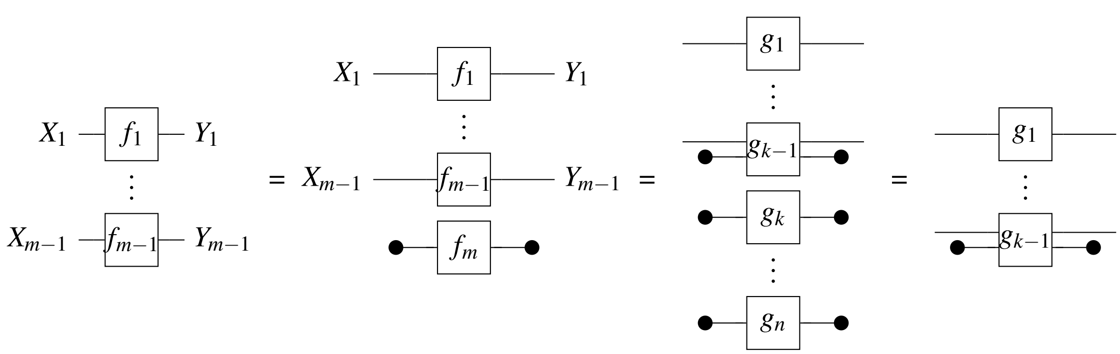
$$\begin{aligned}
&\mathsf{wd}(d')\\
&= \max\{\mathsf{wd}(d'_1), k, \mathsf{wd}(d'_2)\}\\
&\geq k\\
&\geq \mathsf{rank}(\mathsf{Mat} f)\\
&= r_1 + r_2\\
&\geq \max\{r_1, r_2\} + 1\\
&\geq \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2)\}\\
&= \mathsf{wd}(d)
\end{aligned}$$

If there is $r_i = 0$, then $f_i = -\!\bullet \,;_0 \bullet\!-$. We may assume that $f_1 = -\!\bullet \,;_0 \bullet\!-$. Then, $f_2 = (\text{—} \otimes \bullet\!-)\,; g\,;_k h\,; (\text{—} \otimes -\!\bullet)$. By Lemma 3.8, $\mathsf{mwd}((\text{—} \otimes \bullet\!-)\,; g) \leq \mathsf{mwd}(g)$ and $\mathsf{mwd}(h\,; (\text{—} \otimes -\!\bullet)) \leq \mathsf{mwd}(h)$. Then,

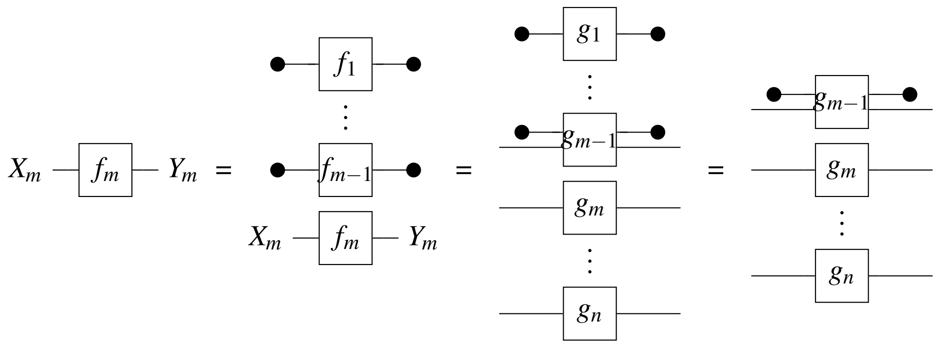
$$\begin{aligned}
&\mathsf{wd}(d')\\
&= \max\{\mathsf{wd}(d'_1), k, \mathsf{wd}(d'_2)\}\\
&\geq \max\{\mathsf{mwd}(g), k, \mathsf{mwd}(h)\}\\
&\geq \max\{\mathsf{mwd}((\text{—} \otimes \bullet\!-)\,; g), k, \mathsf{mwd}(h\,; (\text{—} \otimes -\!\bullet))\}\\
&\geq \mathsf{mwd}(f_2)\\
&= \mathsf{wd}(d_2)\\
&= \mathsf{wd}(d)
\end{aligned}$$

□

Proof sketch of Proposition 3.11. 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. By induction, $\bar{f} := f_1 \otimes \ldots \otimes f_{m-1}$ has a unique $\otimes$-decomposition and we obtain



Then, $k = m-1$, for every $i < m-1$ $f_i = g_i$ and $f_{m-1} = (\text{—} \otimes \bullet\!-)\,; g_k\,; (\text{—} \otimes -\!\bullet)$. With a similar argument, we obtain that $n = m$, $f_{m-1} = g_{m-1}$ and $f_m = g_m$.



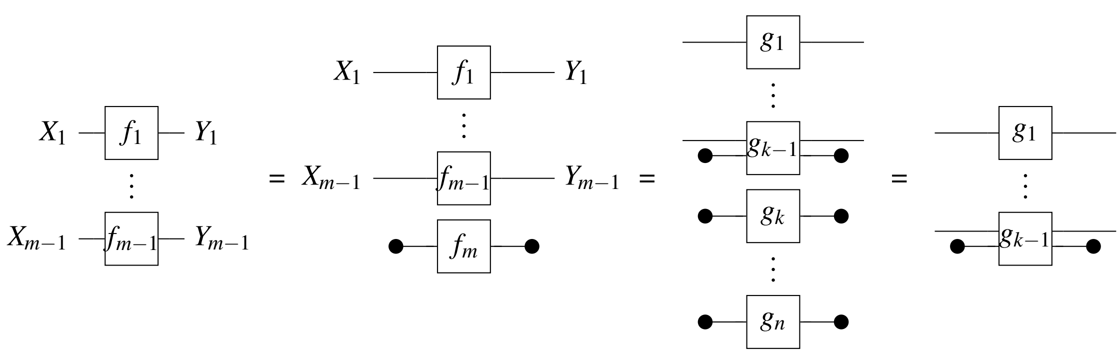
The details of this proof can be found below. □

Proof of Proposition 3.11. 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 $f = \text{—}_0$ and $g_i = \text{—}_0$ for every $i = 1, \dots, n$ because 0 is initial and terminal.

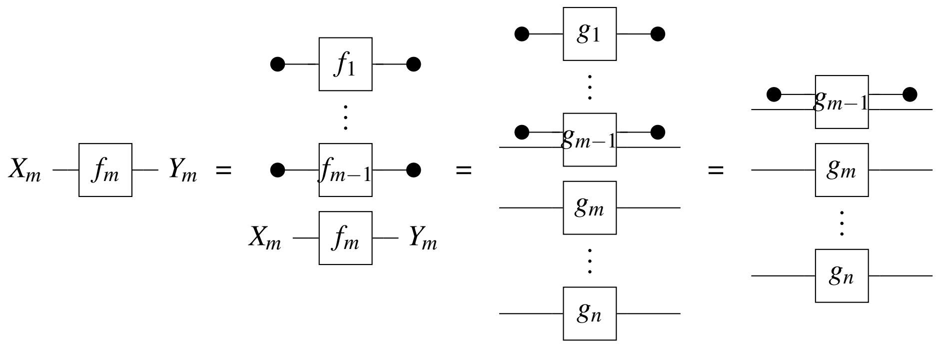
Suppose that $\bar{f} := f_1 \otimes \ldots \otimes f_{m-1}$ has a unique $\otimes$-decomposition. Let $A_1 \otimes \ldots \otimes A_\alpha$ and $B_1 \otimes \ldots \otimes B_\beta$ be the unique $\otimes$-decompositions of $X_1 \otimes \ldots \otimes X_m = Z_1 \otimes \ldots \otimes Z_n$ and $Y_1 \otimes \ldots \otimes Y_m = W_1 \otimes \ldots \otimes W_n$, 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,

$$
\begin{aligned}
&\bar{f} \\
&= (\text{—}_{A_1 \otimes \ldots \otimes A_x} \otimes \bullet\text{–}_{A_{x+1} \otimes \ldots \otimes A_\alpha}) ; f \\
&\quad ; (\text{—}_{B_1 \otimes \ldots \otimes B_y} \otimes \text{–}\bullet_{B_{y+1} \otimes \ldots \otimes B_\beta}) \\
&= (\text{—}_{A_1 \otimes \ldots \otimes A_x} \otimes \bullet\text{–}_{A_{x+1} \otimes \ldots \otimes A_\alpha}) ; (g_1 \otimes \cdots \otimes g_n) \\
&\quad ; (\text{—}_{B_1 \otimes \ldots \otimes B_y} \otimes \text{–}\bullet_{B_{y+1} \otimes \ldots \otimes B_\beta}) \\
&= g_1 \otimes \ldots \otimes g_{k-2} \otimes ((\text{—} \otimes \bullet\text{–}) ; g_{k-1} ; (\text{—} \otimes \text{–}\bullet))
\end{aligned}
$$



By induction hypothesis, it must be that $k = m-1$, for every $i < m-1$ $f_i = g_i$ and $f_{m-1} = (\text{—} \otimes \bullet\text{–}) ; g_k ; (\text{—} \otimes \text{–}\bullet)$. Then,

$$
\begin{aligned}
&f_m \\
&= (\bullet\text{–} \otimes \text{—}_{X_m}) ; f ; (\text{–}\bullet \otimes \text{—}_{Y_m}) \\
&= (\bullet\text{–} \otimes \text{—}_{X_m}) ; (g_1 \otimes \ldots \otimes g_n) ; (\text{–}\bullet \otimes \text{—}_{Y_m}) \\
&= ((\bullet\text{–} \otimes \text{—}) ; g_{m-1} ; (\text{–}\bullet \otimes \text{—})) \otimes g_m \otimes \ldots \otimes g_n
\end{aligned}
$$



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

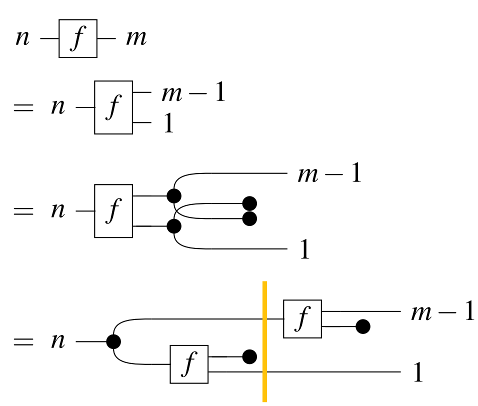
Proof of Proposition 3.5. We proceed by induction on $k = \max\{m,n\}$.

Base cases.

- If $n = 0$, then $f = \bullet\!-_m$ because 0 is initial by hypothesis. Then, $\mathsf{mwd}(f) = \mathsf{mwd}(\bigotimes_m \bullet\!-_1) \leq \mathsf{w}(\bullet\!-_1) \leq 1 \leq 0+1$.
- If $m = 0$, then $f = -\!\bullet_n$ because 0 is terminal by hypothesis. Then, $\mathsf{mwd}(f) = \mathsf{mwd}(\bigotimes_m -\!\bullet_1) \leq \mathsf{w}(-\!\bullet_1) \leq 1 \leq 0+1$.
- If $m = n = 1$, then $\mathsf{mwd}(f) \leq 2 \leq 1+1$ by hypothesis.

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 two possibilities.

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



This corresponds to $f = -\!\!\bullet\!\!\!<_n ; (-\!\!-_n \otimes h_1) ;_{n+1} (h_2 \otimes -\!\!-_1)$, where $h_1 := f ; (-\!\bullet_{m-1} \otimes -\!\!-_1) \colon n \to 1$ and $h_2 := f ; (-\!\!-_{m-1} \otimes -\!\bullet_1) \colon n \to m-1$.

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

$$
\begin{aligned}
&\mathsf{mwd}(\text{-}\!\bullet\!\!\subset_n ; (\text{—}_n \otimes h_1))\\
&\leq \qquad \text{(by Lemma 2.9)}\\
&\max\{\mathsf{mwd}(h_1), n+1\}\\
&\leq \qquad \text{(by induction hypothesis)}\\
&\max\{\min\{n,1\}+1, n+1\}\\
&= \qquad \text{(because } 0 < n\text{)}\\
&= n+1
\end{aligned}
$$

where we could apply the induction hypothesis because $h_1 \colon n \to 1$ and $1, n < k$.

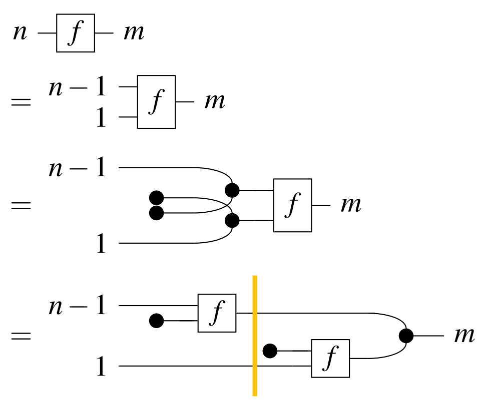
For the second morphism,

$$
\begin{aligned}
&\mathsf{mwd}(h_2 \otimes \text{—}_1)\\
&\leq \qquad \text{(by definition)}\\
&\mathsf{mwd}(h_2)\\
&\leq \qquad \text{(by induction hypothesis)}\\
&\min\{n, m-1\}+1\\
&= \qquad \text{(because } n \leq m-1\text{)}\\
&n+1
\end{aligned}
$$

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

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

2. If $0 < m \leq n = k$, then f can be decomposed as shown below because $\supset\!\!\bullet\!\text{-}_{n+1}$ is uniform by hypothesis and morphisms are cocopiable because P is cocartesian by hypothesis.



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

Then, $\mathsf{mwd}(f) \leq \max\{\mathsf{mwd}(h_1 \otimes \text{—}_1), m+1, \mathsf{mwd}((\text{—}_m \otimes h_2) ; \supset\!\!\bullet\!\text{-}_m)\}$. So, we want to bound the monoidal width of the two morphisms appearing in the formula above. For the first morphism,

$$
\begin{aligned}
&\mathsf{mwd}(h_1 \otimes —_1)\\
&\leq \qquad \text{(by definition)}\\
&\mathsf{mwd}(h_1)\\
&\leq \qquad \text{(by induction hypothesis or point 1)}\\
&\min\{n-1, m\}+1\\
&\leq \qquad \text{(because } m \leq n\text{)}\\
&m+1
\end{aligned}
$$

where, if $m < n$, we could apply the induction hypothesis because $n-1, m < k$, or, if $m = n$, we could apply point 1 because $n-1 < m = k$. For the second morphism,

$$
\begin{aligned}
&\mathsf{mwd}((—_m \otimes h_2)\,;\supset\!\!\bullet\!—_m)\\
&\leq \qquad \text{(by Lemma 2.9)}\\
&\max\{\mathsf{mwd}(h_2), m+1\}\\
&\leq \qquad \text{(by induction hypothesis of point 1)}\\
&\max\{\min\{1, m\}+1, m+1\}\\
&= \qquad \text{(because } m \geq 1\text{)}\\
&m+1
\end{aligned}
$$

where, if $m < n$, we could apply the induction hypothesis because $h_1 \colon n \to 1$ and $1, n < k$, and, if $m = n$, we could apply point 1 because $1 < m = k$.

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

□

Proof of Proposition 3.7. 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.

If $d = (f)$, then either $\mathsf{w}(\asymp) = 2 \geq \mathsf{rank}(\mathsf{Mat}(\asymp)) = 2$, $\mathsf{w}(—\!\bullet\!\!\subset_1) = \mathsf{w}(\supset\!\!\bullet\!—_1) = 2 \geq \mathsf{rank}(\mathsf{Mat}(f)) = 1$ or $\mathsf{w}(—\!\bullet_1) = \mathsf{w}(\bullet\!—_1) = 1 \geq \mathsf{rank}(\mathsf{Mat} f) = 0$. Then, $\mathsf{wd}(d) = \mathsf{w}(f) \geq \mathsf{rank}(\mathsf{Mat}(f))$.

If $d = (d_1, ;_k, d_2)$, then there are $g \colon n \to k$ and $h \colon k \to m$ such that $f = g\,;h$. By Lemma 3.1, $k \geq \mathsf{rank}(\mathsf{Mat}(f))$. Then, $\mathsf{wd}(d) \geq k \geq \mathsf{rank}(\mathsf{Mat}(f))$.

We prove the first inequality. By Lemma 3.1, there are $g \colon n \to r$ and $h \colon r \to m$ such that $f = g\,;h$ with $r = \mathsf{rank}(\mathsf{Mat}(f))$. Then, $r \leq m, n$ by definition of rank. By Lemma 3.6, we can apply Proposition 3.5 to obtan that $\mathsf{mwd}(g) \leq \min\{n, r\}+1 = r+1$ and $\mathsf{mwd}(h) \leq \min\{m, r\}+1 = r+1$. Then, $\mathsf{mwd}(f) \leq \max\{\mathsf{mwd}(g), r, \mathsf{mwd}(h)\} \leq r+1$. □

B Recursive rank width

Proof of Proposition 4.10. Proceed by induction on $|\mathsf{edges}(Y)|$.

If $|\mathsf{edges}(Y)| = 0$, then either $G = \emptyset$ or G has one vertex. In either case, we define $\mathscr{I}(Y, r) := (\Gamma)$ and obtain $\mathsf{wd}(\mathscr{I}(Y, r)) := \mathsf{rank}(B) \leq \mathsf{wd}(Y, r) + \mathsf{rank}(B)$.

If $|\mathsf{edges}(Y)| > 0$, then $Y = Y_1 — Y_2$ with Y_1 and Y_2 subcubic trees. Let $V_i := r(\mathsf{leaves}(Y_i))$ and $G_i := G[V_i]$ be the subgraph of G on the vertices V_i. Since $\{V_1, V_2\}$ is a partition of the vertices of G because (Y,r) is a rank decomposition of G, then we can write $[G] = \left[\begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}\right]$, $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$. By induction hypothesis, there are recursive rank decompositions T_i of $\Gamma_i = ([G_i], B_i)$. Then, $T = (T_1, \Gamma, T_2)$ is a recursive rank decomposition of Γ and, by applying Lemma 4.11,

$$\begin{aligned}
&\mathsf{wd}(T) \\
&:= \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(B)\} \\
&= \max_{T' \leq T} \mathsf{rank}(\mathsf{boundary}(\lambda(T'))) \\
&= \max_{T' \leq T} \mathsf{rank}(A' \mid C') \\
&\leq \max_{T' \leq T} \mathsf{rank}(C') + \mathsf{rank}(B) \\
&= \max_{b \in \mathsf{edges}(Y)} \mathsf{ord}(b) + \mathsf{rank}(B) \\
&=: \mathsf{wd}(Y,r) + \mathsf{rank}(B)
\end{aligned}$$

□

Proof of Lemma 4.11. Proceed by induction on T.

If $T = (\Gamma)$, then $\Gamma' = \emptyset$ or $\Gamma' = \Gamma$ and we are done.

If $T = (T_1, \Gamma, T_2)$, then $\lambda(T_i) = \Gamma_i = ([G_i], B_i)$ 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 $T' \subseteq T_1$. 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, $\mathsf{rank}(B') = \mathsf{rank}(A' \mid E' \mid C_L^\top \mid D_R)$. The rank is invariant to permuting the order of columns, thus $\mathsf{rank}(B') = \mathsf{rank}(A' \mid C_L^\top \mid D_R \mid E') = \mathsf{rank}(A' \mid C_L^\top \mid C_R)$. We proceed analogously if $T' \subseteq T_2$. □

Proof of Proposition 4.12. 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 subgraph of Γ with one vertex v_l. Then, there is a bijection $r\colon \mathsf{leaves}(T) \to \mathsf{vertices}(G)$ such that $r(l) := v_l$. Then, (Y,r) is a branch decomposition of G and we can define $\mathscr{I}^\dagger(T) := (Y,r)$.

By construction, $b \in \mathsf{edges}(Y)$ if and only if $b \in \mathsf{edges}(T)$. Let $\{v, v_b\} = \mathsf{ends}(b)$ with v parent of v_b in T and let T_b the subtree of T with root v_b, with $\lambda(T_b) = \Gamma_b = ([G_b], B_b)$. Then, we can write $[G] = \left[\begin{pmatrix} G_L & C_L & C \\ \mathbb{0} & G_b & C_R \\ \mathbb{0} & \mathbb{0} & G_R \end{pmatrix}\right]$ and $B = \begin{pmatrix} A_L \\ A' \\ A_R \end{pmatrix}$. By Lemma 4.11, $\mathsf{rank}(B_b) = \mathsf{rank}(A' \mid C_L^\top \mid C_R)$. By Definition 4.4, $\mathsf{ord}(b) := \mathsf{rank}(C_L^\top \mid C_R)$. Then, $\mathsf{rank}(B_b) \geq \mathsf{ord}(b)$ and

$$\begin{aligned}
&\mathsf{wd}(Y,r) \\
&:= \max_{b \in \mathsf{edges}(Y)} \mathsf{ord}(b) \\
&\leq \max_{b \in \mathsf{edges}(Y)} \mathsf{rank}(B_b) \\
&\leq \max_{T' \subseteq T} \mathsf{rank}(\mathsf{boundary}(\lambda(T'))) \\
&=: \mathsf{wd}(T)
\end{aligned}$$

□

C Monoidal width and rank width

Proof of Lemma 5.7. Proceed by induction on T.

If $T = (\Gamma)$, then we define $T' := (\Gamma')$. Clearly, T and T' have the same underlying tree structure. Since M has full rank, we can compute $\mathsf{wd}(T') := \mathsf{rank}(B') = \mathsf{rank}(B' \cdot M) =: \mathsf{wd}(T)$.

If $T = (T_1, \Gamma, T_2)$, then $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$, where $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ where $\Gamma_i := \lambda_i(T_i)$, $B_i := \mathsf{boundary}(\Gamma_i)$, $G_i = \mathsf{adjacency}(\Gamma_i)$ and $A_i \in \mathsf{Mat}_{\mathbb{N}}(n, k_i)$. Then, $\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} A'_1 \\ A'_2 \end{pmatrix} \cdot M = \begin{pmatrix} A'_1 \cdot M \\ A'_2 \cdot M \end{pmatrix}$ and $B_i = (A_i \mid C_i) = (A'_i \mid C_i) \cdot \begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_{-i}} \end{pmatrix}$. Let $B'_i := (A'_i \mid C_i)$. The matrix $\begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_{-i}} \end{pmatrix}$ has full rank because both its blocks do. By induction hypothesis, there are recursive rank decompositions T'_i of $\Gamma_i = ([G_i], B_i)$ with the same underlying tree structure as T_i and such that $\mathsf{wd}(T'_i) = \mathsf{wd}(T_i)$. Then, $T' := (T'_1, \Gamma', T'_2)$ is a recursive rank decomposition of Γ' because $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B'_1 = (A'_1 \mid C)$ and $B'_2 = (A'_2 \mid C^\top)$. We can compute

$$\begin{aligned}
&\mathsf{wd}(T') \\
&:= \max\{\mathsf{rank}(B'), \mathsf{wd}(T'_1), \mathsf{wd}(T'_2)\} \\
&= \max\{\mathsf{rank}(B' \cdot M), \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
&=: \mathsf{wd}(T)
\end{aligned}$$

□

Proof of Lemma 5.6. Let us indicate with $C_1 := C$ and $C_2 := C^\top$ and let $L_i \cdot M_i$ be a rank decomposition of $(A_i \mid C_i)$, with $L_i \in \mathsf{Mat}_{\mathbb{N}}(k_i, r_i)$, $M_1 \in \mathsf{Mat}_{\mathbb{N}}(r_1, n + k_2)$ and $M_2 \in \mathsf{Mat}_{\mathbb{N}}(r_2, n + k_1)$. By definition of rank decomposition, L_i and M_i have full rank r_i. We can write $M_i = (N_i \mid K_i)$ with $N_i \in \mathsf{Mat}_{\mathbb{N}}(r_i, n)$, $K_1 \in \mathsf{Mat}_{\mathbb{N}}(r_1, k_2)$ and $K_2 \in \mathsf{Mat}_{\mathbb{N}}(r_2, k_1)$. Then, $C_i = L_i \cdot K_i$. The rank of a composition is the minimum of the ranks and L_i has full rank, then

$$\begin{aligned}
\mathsf{rank}(C) &= \mathsf{rank}(L_1 \cdot K_1) = \mathsf{rank}(K_1) \\
\mathsf{rank}(C) &= \mathsf{rank}(C^\top) = \mathsf{rank}(L_2 \cdot K_2) = \mathsf{rank}(K_2)
\end{aligned}$$

It follows that there are rank decompositions $K_i = P_i \cdot Q_i$ with $Q_i \in \mathsf{Mat}_{\mathbb{N}}(s, r_i)$ and $s = \mathsf{rank}(C)$. Then, $C = (L_1 \cdot P_1) \cdot Q_1 = Q_2^\top \cdot (P_2^\top \cdot L_2^\top)$ are rank factorizations of C. By Lemma 3.1, there is an invertible matrix $R \in \mathsf{Mat}_{\mathbb{N}}(s, s)$ such that $L_1 \cdot P_1 = Q_2^\top \cdot R$ and $Q_1 = R^{-1} \cdot (P_2^\top \cdot L_2^\top)$. Let $S := P_1 \cdot R^{-1} \cdot P_2^\top$ and we obtain $C = L_1 \cdot S \cdot L_2^\top$. Then, $(A_1 \mid C) = L_1 \cdot (N_1 \mid S \cdot L_2^\top)$ and $(A_2 \mid C^\top) = L_2 \cdot (N_2 \mid S^\top \cdot L_1^\top)$. □

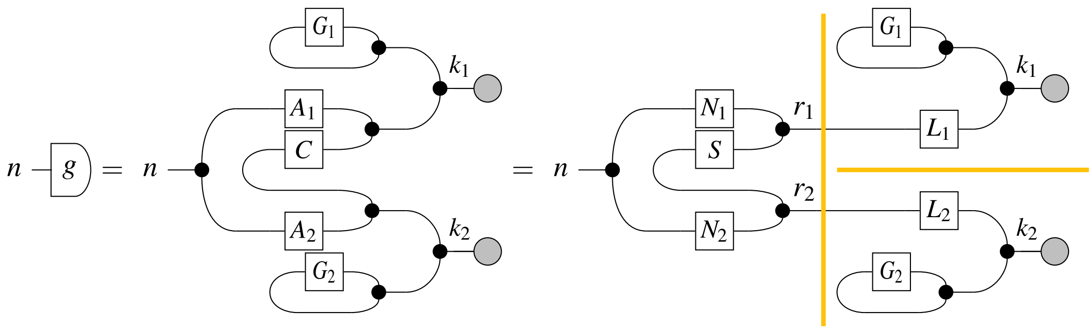
Proof of Proposition 5.8. Proceed by induction on T.

If $T = ()$, then G must be empty, $\mathscr{R}^\dagger(T) = ()$ and we are done.

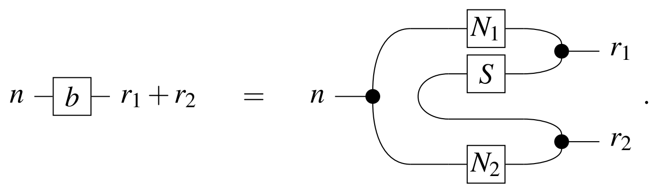
If $T = (\Gamma)$, then Γ has one vertex, we define $\mathscr{R}^\dagger(T) := (g)$ and $\mathsf{wd}(T) := \mathsf{rank}(G) = \mathsf{wd}(\mathscr{R}^\dagger(T))$.

If $T = (T_1, \Gamma, T_2)$, then $\lambda(T_i) = \Gamma_i = ([G_i], B_i)$, with $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$. By Lemma 5.6, 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 $r_i = \mathsf{rank}(B_i)$. Then, $B_i = L_i \cdot M_i$ with M_i that has full rank r_i. By Lemma 5.7, there is a recursive 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 $g_i \colon r_i \to 0$ be the morphisms in Grph corresponding to Γ_i' and let $b \colon n \to r_1 + r_2$ be defined as



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

$$
\begin{aligned}
&\mathsf{wd}(\mathscr{R}^\dagger(T)) \\
&:= \max\{\mathsf{w}(b), \mathsf{w}(r_1+r_2), \mathsf{wd}(\mathscr{R}^\dagger(T_1')), \mathsf{wd}(\mathscr{R}^\dagger(T_2'))\} \\
&\leq \max\{\mathsf{w}(b), \mathsf{w}(r_1+r_2), 2 \cdot \mathsf{wd}(T_1'), 2 \cdot \mathsf{wd}(T_2')\} \\
&= \max\{\mathsf{w}(b), r_1+r_2, 2 \cdot \mathsf{wd}(T_1), 2 \cdot \mathsf{wd}(T_2)\} \\
&\leq 2 \cdot \max\{r_1, r_2, \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
&=: 2 \cdot \mathsf{wd}(T)
\end{aligned}
$$

□

Proof of Lemma 5.10. Proceed by induction on T. If $T = (\Gamma)$, then we define $T' := (\Gamma')$. Clearly, T and T' have the same underlying tree structure. We can compute $\mathsf{wd}(T') := \mathsf{rank}(B') = \mathsf{rank}(B \cdot M) \leq \mathsf{rank}(B) =: \mathsf{wd}(T)$. The inequality becomes an equality when M has full rank and we are done.

If $T = (T_1, \Gamma, T_2)$, then $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B_1 = (A_1 \mid C)$ and $B_2 = (A_2 \mid C^\top)$, where $B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ where $\Gamma_i := \lambda_i(T_i)$, $B_i := \mathsf{boundary}(\Gamma_i)$, $G_i = \mathsf{adjacency}(\Gamma_i)$ and $A_i \in \mathsf{Mat}_{\mathbb{N}}(n, k_i)$. Then, $\begin{pmatrix} A_1' \\ A_2' \end{pmatrix} = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \cdot M = \begin{pmatrix} A_1 \cdot M \\ A_2 \cdot M \end{pmatrix}$ and $B_i' = (A_i' \mid C_i) = (A_i \mid C_i) \cdot \begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_{-i}} \end{pmatrix}$. By induction hypothesis, there are recursive rank decompositions T_i' of $\Gamma_i' = ([G_i], B_i')$ with the same underlying tree structure as T_i and such that $\mathsf{wd}(T_i') \leq \mathsf{wd}(T_i)$. If M has full rank, then $\begin{pmatrix} M & \mathbb{0} \\ \mathbb{0} & \mathbb{1}_{k_{-i}} \end{pmatrix}$ has full rank too because both its blocks do. In this case, the induction hypothesis gives $\mathsf{wd}(T_i') = \mathsf{wd}(T_i)$. Define $T' := (T_1', \Gamma', T_2')$. It is a recursive rank decomposition of Γ'

because $G = \begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}$, $B'_1 = (A'_1 \mid C)$ and $B'_2 = (A'_2 \mid C^\top)$. We can compute

$$\begin{aligned}
&\mathsf{wd}(T') \\
&:= \max\{\mathsf{rank}(B'), \mathsf{wd}(T'_1), \mathsf{wd}(T'_2)\} \\
&= \max\{\mathsf{rank}(B \cdot M), \mathsf{wd}(T'_1), \mathsf{wd}(T'_2)\} \\
&\leq \max\{\mathsf{rank}(B), \mathsf{wd}(T_1), \mathsf{wd}(T_2)\} \\
&=: \mathsf{wd}(T)
\end{aligned}$$

Again, the inequality is an equality when M has full rank. □

Proof of Lemma 5.9. Note that $(L \mid R + L \cdot (F + F^\top) \cdot P^\top) = (L \mid R) \cdot \begin{pmatrix} \mathbb{1}_j & (F+F^\top)\cdot P^\top \\ \mathbb{0} & \mathbb{1}_m \end{pmatrix}$. Then, $\mathsf{rank}(L \mid R + L \cdot (F + F^\top) \cdot P^\top) \leq \mathsf{rank}(L \mid R)$. Proceed by induction on T.

If $T = (\Gamma)$, then Γ has one vertex and we can define $T' := (\Gamma')$. We can compute $\mathsf{wd}(T') := \mathsf{rank}(L \mid R + L \cdot (F + F^\top) \cdot P^\top) \leq \mathsf{rank}(L \mid R) =: \mathsf{wd}(T)$.

If $T = (T_1, \Gamma, T_2)$, then there are $\Gamma_1 = ([G_1], (L_1 \mid R_1 \mid C))$ and $\Gamma_2 = ([G_2], (L_2 \mid R_2 \mid C))$ such that T_i is a recursive rank decomposition of Γ_i, and we can write $[G] = \left[\begin{pmatrix} G_1 & C \\ \mathbb{0} & G_2 \end{pmatrix}\right]$ and $(L \mid R) = \begin{pmatrix} L_1 & R_1 \\ L_2 & R_2 \end{pmatrix}$. We can write $\left[G + L \cdot F \cdot L^\top\right] = \left[\begin{pmatrix} G_1 + L_1 \cdot F \cdot L_1^\top & C + L_1 \cdot (F + F^\top) \cdot L_2^\top \\ \mathbb{0} & G_2 + L_2 \cdot F \cdot L_2^\top \end{pmatrix}\right]$ and $(L \mid R + L \cdot (F + F^\top) \cdot P^\top) = \begin{pmatrix} L_1 & R_1 + L_1 \cdot (F+F^\top) \cdot P^\top \\ L_2 & R_2 + L_2 \cdot (F+F^\top) \cdot P^\top \end{pmatrix}$. Let

$$\begin{aligned}
\Gamma'_1 :=& (\left[G_1 + L_1 \cdot F \cdot L_1^\top\right], (L_1 \mid R_1 + L_1 \cdot (F + F^\top) \cdot P^\top \mid C + L_1 \cdot (F + F^\top) \cdot L_2^\top)) \\
=& (\left[G_1 + L_1 \cdot F \cdot L_1^\top\right], (L_1 \mid (R_1 \mid C) + L_1 \cdot (F + F^\top) \cdot (P^\top \mid L_2^\top)))
\end{aligned}$$

and

$$\begin{aligned}
\Gamma'_2 :=& (\left[G_2 + L_2 \cdot F \cdot L_2^\top\right], (L_2 \mid R_2 + L_2 \cdot (F + F^\top) \cdot P^\top \mid C^\top + L_2 \cdot (F + F^\top) \cdot L_1^\top)) \\
=& (\left[G_2 + L_2 \cdot F \cdot L_2^\top\right], (L_2 \mid (R_2 \mid C^\top) + L_2 \cdot (F + F^\top) \cdot (P^\top \mid L_1^\top)))
\end{aligned}$$

By induction, we have recursive rank decompositions T'_i of Γ'_i such that $\mathsf{wd}(T'_i) \leq \mathsf{wd}(T_i)$. Since Γ'_i satisfy the conditions for a recursive rank decomposition, we can define a recursive rank decomposition of Γ' as $T' := (T'_1, \Gamma', T'_2)$, and compute its width.

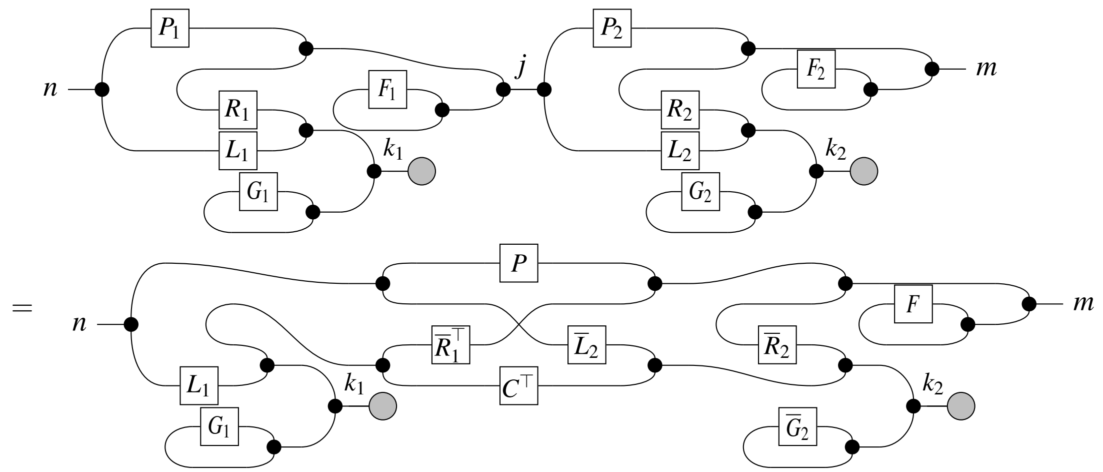
$$\begin{aligned}
&\mathsf{wd}(T') \\
&:= \max\{\mathsf{wd}(T'_1), \mathsf{wd}(T'_2), \mathsf{rank}(L \mid R + L \cdot (F + F^\top) \cdot P^\top)\} \\
&\leq \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(L \mid R + L \cdot (F + F^\top) \cdot P^\top)\} \\
&\leq \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(L \mid R)\} \\
&=: \mathsf{wd}(T)
\end{aligned}$$

□

Proof of Proposition 5.11. Proceed by induction on d.

If $d = (g)$, then $\mathsf{wd}(d) := k$, where k is the number of vertices of g. Pick any recursive rank decomposition of Γ and define $\mathscr{R}(d) := T$. Surely, $\mathsf{wd}(T) \leq k =: \mathsf{wd}(d)$

If $d = (d_1, ;_j, d_2)$, then there are $g_i = ([G_i], L_i, R_i, P_i, [F_i])$ such that $g = g_1 ; g_2$. Given the partition of the vertices determined by g_1 and g_2, we can decompose g in another way, by writing $[G] = \left[\begin{pmatrix} \overline{G}_1 & C \\ \mathbb{0} & \overline{G}_2 \end{pmatrix}\right]$ and $B = (L \mid R) = \begin{pmatrix} \overline{L}_1 & \overline{R}_1 \\ \overline{L}_2 & \overline{R}_2 \end{pmatrix}$. 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 $F = F_2 + P_2 \cdot F_1 \cdot P_2^\top$. Diagrammatically, this corresponds to the following equation.



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 a recursive rank decomposition of Γ, we need recursive rank decompositions of $\overline{\Gamma}_i = (\left[\overline{G}_i\right], \overline{B}_i)$. We obtain these in three steps. Firstly, we apply induction to obtain recursive rank decompositions $\mathscr{R}(d_i)$ of $\Gamma_i = ([G_i], (L_i \mid R_i))$ such that $\mathsf{wd}(\mathscr{R}(d_i)) \leq 2 \cdot \max\{\mathsf{wd}(d_i), \mathsf{rank}(L_i), \mathsf{rank}(R_i)\}$. Secondly, we apply Lemma 5.9 to obtain a recursive 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 $\mathsf{wd}(T_2') \leq \mathsf{wd}(\mathscr{R}(d_2))$. 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 $\overline{B}_1 = (L_1 \mid R_1) \cdot \begin{pmatrix} \mathbb{1}_n & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & P_2^\top & L_2^\top \end{pmatrix}$ and $\overline{B}_2 = (L_2 \mid R_2 + L_2 \cdot (F_1 + F_1^\top) \cdot P_2^\top) \cdot \begin{pmatrix} P_1 & \mathbb{0} & R_1^\top \\ \mathbb{0} & \mathbb{1}_m & \mathbb{0} \end{pmatrix}$, and we can apply Lemma 5.10 to get recursive rank decompositions T_i of $\overline{\Gamma}_i$ such that $\mathsf{wd}(T_1) \leq \mathsf{wd}(\mathscr{R}(d_1))$ and $\mathsf{wd}(T_2) \leq \mathsf{wd}(T_2') \leq \mathsf{wd}(\mathscr{R}(d_2))$. If $k_1, k_2 > 0$, then we define $\mathscr{R}(d) := (T_1, \Gamma, T_2)$, which is a recursive rank decomposition of Γ because $\overline{\Gamma}_i$ satisfy the two conditions of a recursive rank decomposition from Definition 4.8. If $k_1 = 0$, then $\Gamma = \overline{\Gamma}_2$ and we can define $\mathscr{R}(d) := T_2$. Similarly, if $k_2 = 0$, then $\Gamma = \overline{\Gamma}_1$ and we can define $\mathscr{R}(d) := T_1$. In any case, we can compute the width of $\mathscr{R}(d)$ (if $k_i = 0$ then $T_i = ()$ and $\mathsf{wd}(T_i) = 0$) using the inductive hypothesis, Lemma 5.9, Lemma 5.10, the fact that $\mathsf{rank}(L) \geq \mathsf{rank}(L_1)$, $\mathsf{rank}(R) \geq \mathsf{rank}(R_2)$ and $j \geq \mathsf{rank}(R_1), \mathsf{rank}(L_2)$ because $R_1 \colon j \to k_1$ and $L_2 \colon j \to k_2$.

$$
\begin{aligned}
&\mathsf{wd}(T) \\
&:= \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(L \mid R)\} \\
&\leq \max\{\mathsf{wd}(\mathscr{R}(d_1)), \mathsf{wd}(T_2'), \mathsf{rank}(L \mid R)\} \\
&\leq \max\{\mathsf{wd}(\mathscr{R}(d_1)), \mathsf{wd}(\mathscr{R}(d_2)), \mathsf{rank}(L \mid R)\} \\
&\leq \max\{\mathsf{wd}(\mathscr{R}(d_1)), \mathsf{wd}(\mathscr{R}(d_2)), \mathsf{rank}(L) + \mathsf{rank}(R)\} \\
&\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot \mathsf{rank}(L_1), 2 \cdot \mathsf{rank}(R_1), 2 \cdot \mathsf{wd}(d_2), 2 \cdot \mathsf{rank}(L_2), 2 \cdot \mathsf{rank}(R_2), \mathsf{rank}(L) + \mathsf{rank}(R)\} \\
&\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{rank}(L_1), \mathsf{rank}(R_1), \mathsf{wd}(d_2), \mathsf{rank}(L_2), \mathsf{rank}(R_2), \mathsf{rank}(L), \mathsf{rank}(R)\} \\
&\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2), j, \mathsf{rank}(L), \mathsf{rank}(R)\} \\
&=: 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rank}(L), \mathsf{rank}(R)\}
\end{aligned}
$$

If $d = (d_1, \otimes, d_2)$, then there are $g_i = ([G_i], L_i, R_i, P_i, [F_i]) \colon n_i \to m_i$ such that $g = g_1 \otimes g_2$. By exlicitly computing the monoidal product, we obtain that $[G] = \left[\begin{pmatrix} G_1 & \mathbb{0} \\ \mathbb{0} & G_2 \end{pmatrix}\right]$, $L = \begin{pmatrix} L_1 & \mathbb{0} \\ \mathbb{0} & L_2 \end{pmatrix}$, $R = \begin{pmatrix} R_1 & \mathbb{0} \\ \mathbb{0} & R_2 \end{pmatrix}$, $P = \begin{pmatrix} P_1 & \mathbb{0} \\ \mathbb{0} & P_2 \end{pmatrix}$ and $F = \begin{pmatrix} F_1 & \mathbb{0} \\ \mathbb{0} & F_2 \end{pmatrix}$. By induction, we have recursive rank decompositions $\mathscr{R}(d_i)$ of $\Gamma_i := ([G_i], B_i)$, where $B_i = (L_i \mid R_i)$, such that $\mathsf{wd}(\mathscr{R}(d_i)) \leq 2 \cdot \max\{\mathsf{wd}(d_i), \mathsf{rank}(L_i), \mathsf{rank}(R_i)\}$. Let $\overline{B}_1 := (L_1 \mid \mathbb{0}_{n_2} \mid R_1 \mid \mathbb{0}_{m_2} \mid \mathbb{0}_{k_2}) = B_1 \cdot \begin{pmatrix} \mathbb{1}_{n_1} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{m_1} & \mathbb{0} & \mathbb{0} \end{pmatrix}$ and $\overline{B}_2 := (\mathbb{0}_{n_1} \mid L_2 \mid \mathbb{0}_{m_1} \mid R_2 \mid \mathbb{0}_{k_1}) = B_2 \cdot \begin{pmatrix} \mathbb{0} & \mathbb{1}_{n_2} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{m_2} & \mathbb{0} \end{pmatrix}$. By Lemma 5.10, we can obtain recursive rank decompositions T_i of $\overline{\Gamma}_i := ([G_i], \overline{B}_i)$ such that $\mathsf{wd}(T_i) \leq \mathsf{wd}(\mathscr{R}(d_i))$. If $k_1, k_2 > 0$, then we define $\mathscr{R}(d) := (T_1, \Gamma, T_2)$, which is a recursive rank decomposition of Γ because $\overline{\Gamma}_i$ satisfy the two conditions of a recursive rank decomposition in Definition 4.8. If $k_1 = 0$, then $\Gamma = \overline{\Gamma}_2$ and we can define $\mathscr{R}(d) := T_2$. Similarly, if $k_2 = 0$, then $\Gamma = \overline{\Gamma}_1$ and we can define $\mathscr{R}(d) := T_1$. In any case, we can compute the width of $\mathscr{R}(d)$ (if $k_i = 0$ then $T_i = ()$ and $\mathsf{wd}(T_i) = 0$) using the inductive hypothesis and Lemma 5.10.

$$
\begin{aligned}
&\mathsf{wd}(T) \\
&:= \max\{\mathsf{wd}(T_1), \mathsf{wd}(T_2), \mathsf{rank}(L \mid R)\} \\
&\leq \max\{\mathsf{wd}(\mathscr{R}(d_1)), \mathsf{wd}(\mathscr{R}(d_2)), \mathsf{rank}(L \mid R)\} \\
&\leq \max\{\mathsf{wd}(\mathscr{R}(d_1)), \mathsf{wd}(\mathscr{R}(d_2)), \mathsf{rank}(L) + \mathsf{rank}(R)\} \\
&\leq \max\{2 \cdot \mathsf{wd}(d_1), 2 \cdot \mathsf{rank}(L_1), 2 \cdot \mathsf{rank}(R_1), 2 \cdot \mathsf{wd}(d_2), 2 \cdot \mathsf{rank}(L_2), 2 \cdot \mathsf{rank}(R_2), \mathsf{rank}(L) + \mathsf{rank}(R)\} \\
&\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{rank}(L_1), \mathsf{rank}(R_1), \mathsf{wd}(d_2), \mathsf{rank}(L_2), \mathsf{rank}(R_2), \mathsf{rank}(L), \mathsf{rank}(R)\} \\
&\leq 2 \cdot \max\{\mathsf{wd}(d_1), \mathsf{wd}(d_2), \mathsf{rank}(L), \mathsf{rank}(R)\} \\
&=: 2 \cdot \max\{\mathsf{wd}(d), \mathsf{rank}(L), \mathsf{rank}(R)\}
\end{aligned}
$$

□