lomonads meetups

25 May 2022

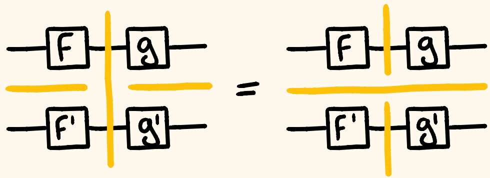
# MONOIDAL WIDTH

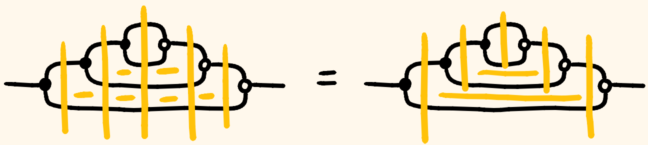

Elena Di Lavore and Paweł Sobociński

Tallinn University of Technology

# MOTIVATION (1)

- how efficient is to compute the semantics of morphisms in monoidal categories?




- we need an 'algebra of decompositions'

# MOTIVATION (2)

- existing notions of complexity for graphs are based on decompositions : path width, tree width, branch width and rank width

- make explicit the algebra of decomposition that is hidden behind the definitions of these graph widths

# MAIN RESULTS

- monoidal width as a measure of complexity for morphisms in monoidal categories
- monoidal decomposition as explicit decomposition algebra
- capture some known measures of complexity for graphs: path width, tree width, branch width and rank width

# OUTLINE

- monoidal decompositions
- monoidal width for matrices
- monoidal width for path, tree & branch width
- monoidal width for rank width

# DECOMPOSITION SYSTEM

A decomposition system $(\mathcal{A}, \mathcal{O}, w)$ in a monoidal category $\mathcal{C}$ is given by

- $\mathcal{A}$ : set of 'atomic' morphisms in $\mathcal{C}$
- $\mathcal{O} = \{\otimes, ;_X \text{ for } X \in obj(\mathcal{C})\}$ : set of operations
- $w : \mathcal{A} \cup \mathcal{O} \to \mathbb{N}$ : weight function

  such that:

$$\begin{cases} w(\otimes) = 0 \\ w(;_{X\otimes Y}) = w(;_X) + w(;_Y) \end{cases}$$

# DECOMPOSITION SYSTEM - EXAMPLE

a decomposition system $(\mathcal{A}, \mathcal{O}, w)$

in $\mathcal{C}$ $\rightsquigarrow$ $\mathcal{F}in\mathcal{S}et$

- $\mathcal{A}$ : set of 'atoms' $\rightsquigarrow$ $\{ \supset\!-, \bullet\!-, \times, - \}$

- $\mathcal{O} = \{ \otimes, ;_X \text{ for } X \in obj(\mathcal{C}) \}$ : set of operations

- $w : \mathcal{A} \cup \mathcal{O} \longrightarrow \mathbb{N}$ : weight $\rightsquigarrow$ $w(\supset\!-) = w(\times) = 2$
  $w(\bullet\!-) = w(-) = 1$
  $w(;_n) = n$

  such that:

$$\begin{cases} w(\otimes) = 0 \\ w(;_{X \otimes Y}) = w(;_X) + w(;_Y) \end{cases}$$

# MONOIDAL DECOMPOSITION

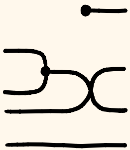
$f : X \to Y$ morphism in $\mathcal{C}$

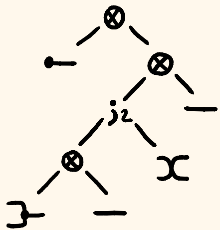
a monoidal decomposition $d \in \mathcal{D}_f$ of $f$ is

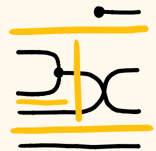
$$
\begin{array}{llll}
d ::= & (f) & \text{if } f \in \mathcal{A} \\
\mid & d_1 \overset{;_{\mathcal{C}}}{\wedge} d_2 & \text{if } f = f_1 ;_{\mathcal{C}} f_2 , \; d_1 \in \mathcal{D}_{f_1} , \; d_2 \in \mathcal{D}_{f_2} \\
\mid & d_1 \overset{\otimes}{\wedge} d_2 & \text{if } f = f_1 \otimes f_2 , \; d_1 \in \mathcal{D}_{f_1} , \; d_2 \in \mathcal{D}_{f_2}
\end{array}
$$

$\rightsquigarrow$ it's a labelled binary tree

# MONOIDAL DECOMPOSITION - EXAMPLE

 $: 4 \longrightarrow 4$ morphism in $\mathcal{F}in\mathcal{S}et$

$d =$ 

$\rightsquigarrow$ 

# MONOIDAL WIDTH

$d \in \mathcal{D}_f$ monoidal decomposition of $f$

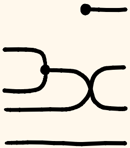
## WIDTH OF $d$

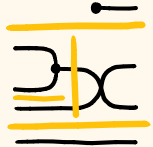
$$
\begin{aligned}
\mathrm{wd}(d) := \; & w(f) && \text{if } d = (f) \\
\mid \; & \max\{\mathrm{wd}(d_1), w(;_c), \mathrm{wd}(d_2)\} && \text{if } d = d_1 \;\overset{;_c}{\wedge}\; d_2 \\
\mid \; & \max\{\mathrm{wd}(d_1), \mathrm{wd}(d_2)\} && \text{if } d = d_1 \;\overset{\otimes}{\wedge}\; d_2
\end{aligned}
$$

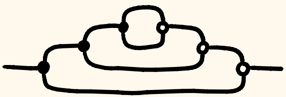
## MONOIDAL WIDTH OF $f$

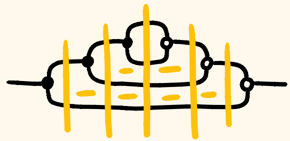
$$\mathrm{mwd}(f) := \min_{d \in \mathcal{D}_f} \mathrm{wd}(d)$$

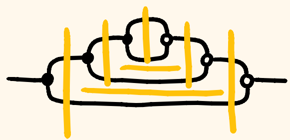
# MONOIDAL WIDTH - EXAMPLE

$f =$ 

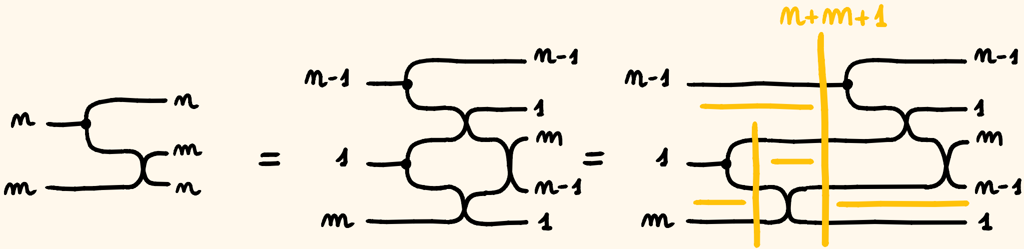
wd(  ) $= 2$

$4 =$ 

wd(  ) $= 4$

wd(  ) $= 2$

# MONOIDAL WIDTH OF COPYING




$m = 0$

$\Rightarrow \mathrm{mwd}(\mathrm{copy}_n) \leq n + 1$

# MONOIDAL TREE DECOMPOSITION

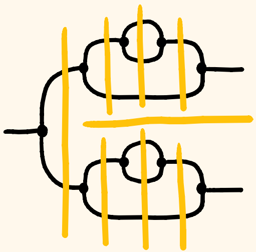
$\rightsquigarrow$ restrict compositions to have an atom on one side and recursion on the other

$f : X \to Y$ in $\mathcal{C}$

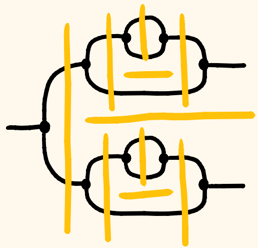
a monoidal tree decomposition $d \in \mathcal{D}^T_f$ of $f$ is

$$
\begin{aligned}
d ::= \; & (f) && \text{if } f \in \mathcal{A} \\
| \; & (g) \frown^{;_c} d' && \text{if } f = g ;_c f' ,\; g \in \mathcal{A} ,\; d' \in \mathcal{D}^T_{f'} \\
| \; & d_1 \frown^{\otimes} d_2 && \text{if } f = f_1 \otimes f_2 ,\; d_1 \in \mathcal{D}^T_{f_1} ,\; d_2 \in \mathcal{D}^T_{f_2}
\end{aligned}
$$

# MONOIDAL TREE DECOMPOSITION - EXAMPLE



✓



✗

# MONOIDAL PATH WIDTH

$\rightsquigarrow$ ban monoidal products completely

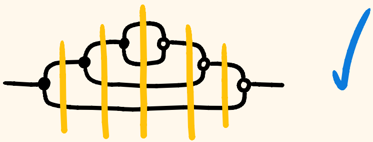
$f : X \to Y$ in $\mathcal{C}$

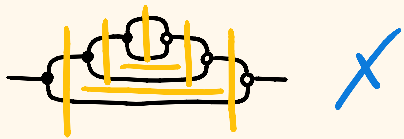
a monoidal path decomposition $d \in \mathcal{D}^p_f$ of $f$ is

$$d ::= (f) \qquad \text{if } f \in \mathcal{A}$$

$$\mid \; d_1 \overset{;c}{\frown} d_2 \qquad \text{if } f = f_1 ;_c f_2 \,,\; d_1 \in \mathcal{D}^p_{f_1} ,\; d_2 \in \mathcal{D}^p_{f_2}$$

# MONOIDAL PATH WIDTH - EXAMPLE

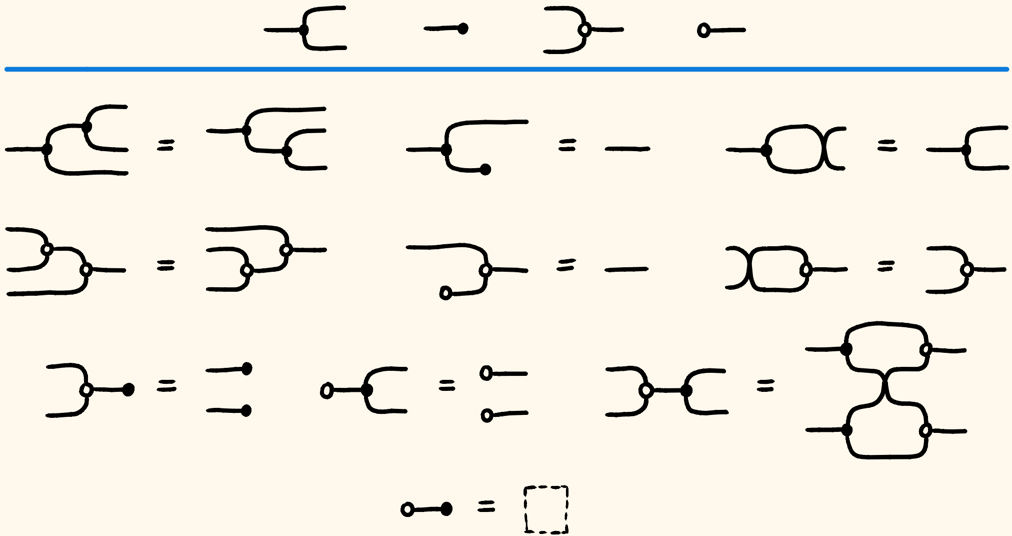




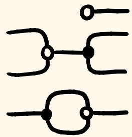
# OUTLINE

- monoidal decompositions
- monoidal width for matrices
- monoidal width for path, tree & branch width
- monoidal width for rank width

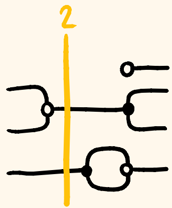
# PROP OF MATRICES



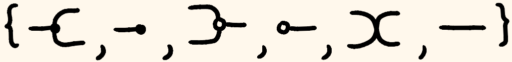
# PROP OF MATRICES - EXAMPLE

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} =$$ 

FACT : the minimal vertical cut in a matrix is its rank : $\min\{k \in \mathbb{N} \mid A = B ;_k C\} = \operatorname{rank} A$

$\operatorname{rank} A = 2$ ⇝ 

# MONOIDAL WIDTH OF MATRICES

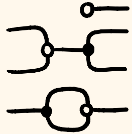
$\mathcal{A} =$ 

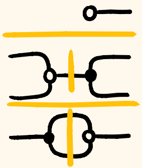
$$A = \begin{pmatrix} A_1 & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & A_2 & & \vdots \\ \vdots & & \ddots & \\ \mathbb{0} & \cdots & & A_b \end{pmatrix} = A_1 \oplus A_2 \oplus \cdots \oplus A_b$$

THEOREM

$$\max_i \operatorname{rank} A_i \leq \operatorname{mwd} A \leq \max_i \operatorname{rank} A_i + 1$$

# MONOIDAL WIDTH OF MATRICES - EXAMPLE

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} =$$ 

$\mathrm{wd}\Bigg($  $\Bigg) = 2$

$$= \max\{\underbrace{\mathrm{rank}(j)}_{=0},\ \underbrace{\mathrm{rank}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}}_{=1},\ \underbrace{\mathrm{rank}(2)}_{=1}\} + 1$$

# OUTLINE

- monoidal decompositions
- monoidal width for matrices
- [ monoidal width for path, tree & branch width ]
- monoidal width for rank width

# PATH WIDTH [Robertson & Seymour, 1983]

$G = (V, E, \mathrm{ends} : E \to \mathcal{P}_{\leq 2}(V))$ undirected graph

## PATH DECOMPOSITION

$(P, p)$ where

- $P = \overset{1}{\bullet} - \overset{2}{\bullet} - \cdots \overset{n}{\bullet}$ is a path
- $p : \mathrm{vertices}\, P \longrightarrow \mathcal{P}(V)$ labelling function

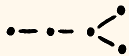
such that

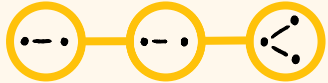
- $\bigcup \{ p(i) : i \in \mathrm{vertices}\, P \} = V$
- $\forall e \in E \;\; \exists i \in \mathrm{vertices}\, P \;\; \mathrm{ends}(e) \subseteq p(i)$

$\leadsto$ cover all the graph

- $\forall\, i < j < k \in \mathrm{vertices}\, P \;\; p(i) \cap p(k) \subseteq p(j)$ $\leadsto$ path shape

# PATH WIDTH - EXAMPLE

$G =$ 

$(P,p) =$ 

WIDTH OF $(P,p)$

Robertson & Seymour

$$wd(P,p) := \max_{i \in \text{vertices } P} |p(i)| \quad (-1) \qquad \rightsquigarrow \quad wd(P,p) = 3$$

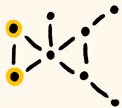
PATH WIDTH

$$pwd(G) := \min_{(P,p)} wd(P,p) \qquad \rightsquigarrow \quad pwd(G) = 3$$

# GRAPHS WITH SOURCES

$G = (V, E)$ undirected graph

$\Gamma = (G, X)$ graph with sources $X \subseteq V$

$\Gamma =$ 

$\rightsquigarrow$ graphs can be 'glued' along their sources

# PATH DECOMPOSITIONS – RECURSIVELY

$\Gamma = ((V, E), X)$ graph with sources

## RECURSIVE PATH DECOMPOSITION

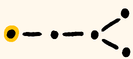
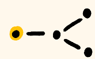
$$T ::= () \qquad \text{if } \Gamma = \emptyset$$

$$\mid (V_1, T') \qquad \text{if } T' \text{ rec. path dec. of } \Gamma' = ((V', E'), X')$$

$$V = V_1 \cup V', \quad X \subseteq V_1$$

$$X' = V_1 \cap V', \quad ends(E \setminus E') \subseteq V_1$$

$\rightsquigarrow$ $\Gamma$ is obtained by 'glueing' $\Gamma'$ with $\Gamma_1 := ((V_1, E \setminus E'), X \cup X')$

 =  'glued' with 

# PATH WIDTH – RECURSIVELY

$T$ recursive path decomposition of $\Gamma = (G, X)$

WIDTH OF $T$

$$wd(T) := \begin{cases} 0 & \text{if } T = () \\ \max\{|V_1|, wd(T')\} & \text{if } T = (V_1, T') \end{cases}$$

RECURSIVE PATH WIDTH

$$rpwd(\Gamma) := \min_{T} wd(T)$$

PROPOSITION

$$rpwd(\Gamma) = pwd(G)$$

# TREE WIDTH [Robertson & Seymour, 1986]

$G = (V, E, \text{ends} : E \to \mathcal{P}_{\leq 2}(V))$ undirected graph

## TREE DECOMPOSITION

$(Y, t)$ where

- $Y$ is a tree (= connected acyclig graph)
- $t : \text{vertices } Y \longrightarrow \mathcal{P}(V)$ labelling function

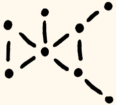
such that

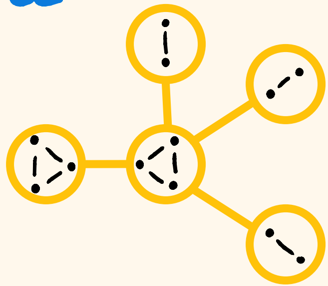
- $\bigcup \{ t(i) : i \in \text{vertices } Y \} = V$
- $\forall e \in E \ \exists i \in \text{vertices } Y \ \ \text{ends}(e) \subseteq t(i)$

$\rightsquigarrow$ cover all the graph

- $\forall \, i \rightsquigarrow j \rightsquigarrow k \in \text{vertices } Y \ \ t(i) \cap t(k) \subseteq t(j)$ $\rightsquigarrow$ tree shape

# TREE WIDTH - EXAMPLE

$G =$ 

$(Y,t) =$ 

WIDTH OF $(Y,t)$

Robertson & Seymour

$wd(Y,t) := \max_{i \in \text{vertices } Y} |t(i)|$ $(-1)$ $\rightsquigarrow$ $wd(Y,t) = 3$

TREE WIDTH

$twd(G) := \min_{(Y,t)} wd(Y,t)$ $\rightsquigarrow$ $twd(G) = 3$

# TREE DECOMPOSITIONS – RECURSIVELY

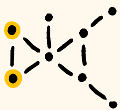
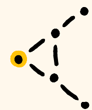
$\Gamma = ((V, E), X)$ graph with sources

## RECURSIVE TREE DECOMPOSITION

$$T ::= () \qquad \text{if } \Gamma = \emptyset$$

$$\mid \quad \begin{array}{ccc} & V' & \\ / & & \backslash \\ T_1 & & T_2 \end{array} \qquad \begin{array}{l} \text{if } T_i \text{ rec. tree dec. of } \Gamma_i = ((V_i, E_i), X_i) \\ V = V_1 \cup V' \cup V_2,\ X \subseteq V' \\ X_i = V_i \cap V',\ V_1 \cap V_2 \subseteq V' \\ E_1 \cap E_2 = \emptyset,\ \mathrm{ends}(E \setminus (E_1 \sqcup E_2)) \subseteq V' \end{array}$$

$\leadsto$ $\Gamma$ is obtained by 'glueing' $\Gamma_1$ and $\Gamma_2$ with $\Gamma' := ((V', E \setminus (E_1 \sqcup E_2)), X \cup X_1 \cup X_2)$

 =  'glued' with  and 

# TREE WIDTH – RECURSIVELY

$T$ recursive tree decomposition of $\Gamma = (G, X)$

## WIDTH OF $T$

$$\mathrm{wd}(T) := \begin{cases} 0 & \text{if } T = () \\ \max\{\mathrm{wd}(T_1), |V'|, \mathrm{wd}(T_2)\} & \text{if } T = \begin{array}{ccc} & V' & \\ T_1 & & T_2 \end{array} \end{cases}$$

## RECURSIVE TREE WIDTH

$$\mathrm{rtwd}(\Gamma) := \min_{T} \mathrm{wd}(T)$$

## PROPOSITION

$$\mathrm{rtwd}(\Gamma) = \mathrm{twd}(G)$$

# BRANCH WIDTH [Robertson & Seymour, 1991]

$G = (V, E, \mathrm{ends} : E \to \mathcal{P}_{\leq 2}(V))$ undirected graph

## BRANCH DECOMPOSITION

$(Y, \ell)$ where

- $Y$ is a subcubic tree (= any node has at most 3 neighbours)
- $\ell$ : leaves $Y \xrightarrow{\simeq} E$ labelling bijection

## WIDTH OF $(Y, \ell)$

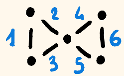
$$\mathrm{wd}(Y, \ell) := \max_{e \in \mathrm{edges}\, Y} |\mathrm{ends}\, A_e \cap \mathrm{ends}\, B_e|$$

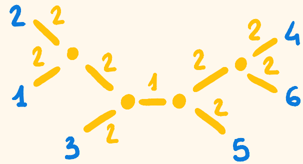
$\{A_e, B_e\}$ partition of $E$ given by $e$ through $\ell$

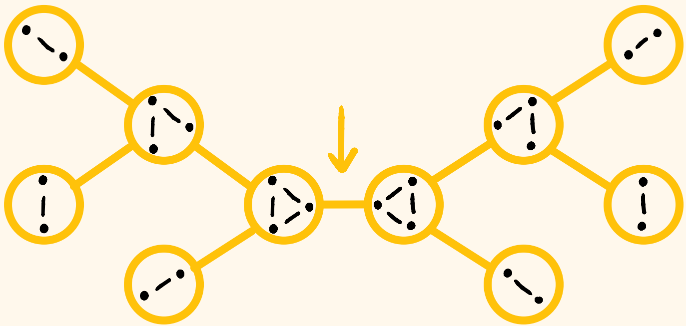
## BRANCH WIDTH

$$\mathrm{bwd}(G) := \min_{(Y, \ell)} \mathrm{wd}(Y, \ell)$$

# BRANCH WIDTH – EXAMPLE

$G =$ 


$(Y, b) =$ 

# BRANCH DECOMPOSITIONS – RECURSIVELY

$\Gamma = ((V, E), X)$ graph with sources

## RECURSIVE BRANCH DECOMPOSITION

$T ::= ()$ if $\Gamma$ is discrete

$\mid (\Gamma)$ if $\Gamma$ has one edge

$\mid \begin{array}{ccc} & \Gamma & \\ T_1 & & T_2 \end{array}$ if $T_i$ rec. branch dec. of $\Gamma_i = ((V_i, E_i), X_i)$
$V = V_1 \cup V_2$, $E = E_1 \sqcup E_2$
$X_i = (V_1 \cap V_2) \cup (V_i \cap X)$

$\rightsquigarrow$ $\Gamma$ is obtained by 'glueing' $\Gamma_1$ and $\Gamma_2$

 =  'glued' with 

# BRANCH WIDTH – RECURSIVELY

$T$ recursive branch decomposition of $\Gamma = (G, X)$

## WIDTH OF $T$

$$
\begin{aligned}
wd(T) &:= 0 && \text{if } T = () \\
&\mid \max\{wd(T_1), |X|, wd(T_2)\} && \text{if } T = \begin{array}{ccc} & \Gamma & \\ / & & \backslash \\ T_1 & & T_2 \end{array}
\end{aligned}
$$

## RECURSIVE BRANCH WIDTH

$$rbwd(\Gamma) := \min_T wd(T)$$

## PROPOSITION

$$bwd(G) \leq rbwd(\Gamma) \leq bwd(G) + |X|$$

# COSPANS OF GRAPHS

$\mathrm{Cospan}(\mathrm{UGraph})_\emptyset$

objects : sets $\rightsquigarrow$ discrete graphs

morphisms $X \to Y$ : cospans $X \xrightarrow{\partial_X} G \xleftarrow{\partial_Y} Y$ of graphs

composition : by pushout $\rightsquigarrow$ glue along vertices

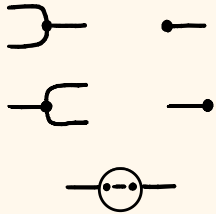
monoidal product : component-wise disjoint union

$\rightsquigarrow$ graphs with left and right sources

# DECOMPOSITIONS IN COSPANS OF GRAPHS

Frobenius structure

+ 'edge' generator



## ATOMS

$$\mathcal{A} = \{\text{all morphisms}\}$$

## WEIGHT FUNCTION

$$w\left( X \rightarrow (V,E) \leftarrow Y \right) := |V|$$

$$w(j_X) := |X|$$

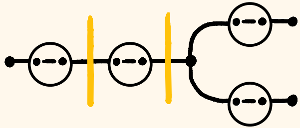
# PATH WIDTH & MONOIDAL WIDTH

$G = (V, E)$ undirected graph

$g = \emptyset \xrightarrow{i} G \xleftarrow{i} \emptyset : \emptyset \to \emptyset$ in $\mathrm{Cospan}(\mathrm{UGraph})_\emptyset$

THEOREM

$$\mathrm{pwd}(G) = \mathrm{mpwd}(g)$$

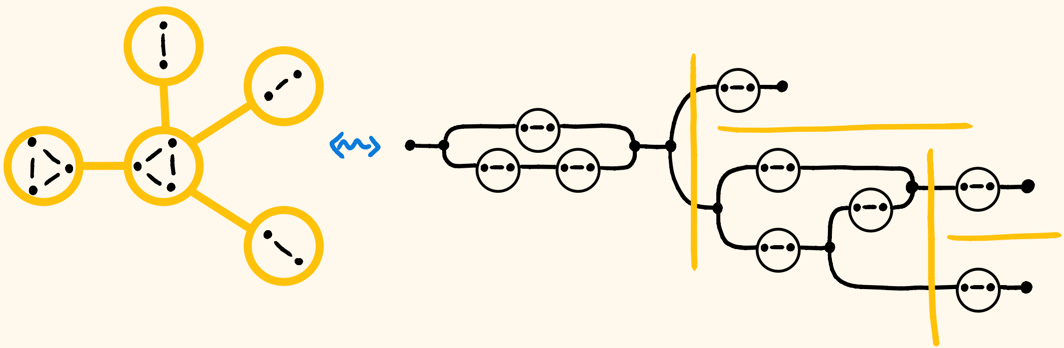


# TREE WIDTH & MONOIDAL WIDTH

$G = (V, E)$ undirected graph
$g = \emptyset \xrightarrow{i} G \xleftarrow{i} \emptyset : \emptyset \to \emptyset$ in $\mathsf{cospan}(\mathsf{UGraph})_\emptyset$

THEOREM

$$\mathrm{twd}(G) \leq \mathrm{mtwd}(g) \leq 2 \cdot \mathrm{twd}(G)$$



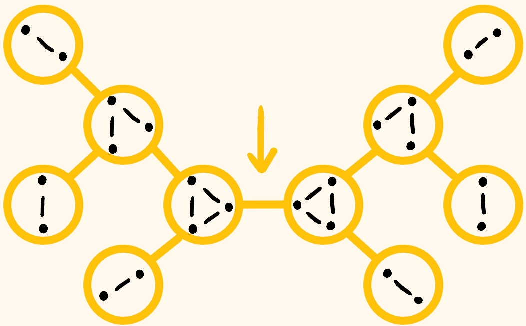
# BRANCH WIDTH & MONOIDAL WIDTH

$G = (V, E)$ undirected graph

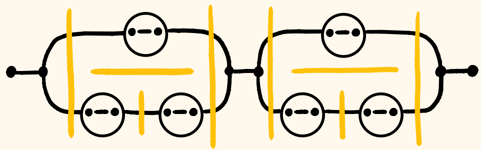
$g = \emptyset \xrightarrow{i} G \xleftarrow{i} \emptyset : \emptyset \to \emptyset$ in $\mathrm{cospan}(\mathrm{UGraph})_{\emptyset}$

THEOREM

$$\frac{1}{2}\, \mathrm{bwd}(G) \leq \mathrm{mwd}(g) \leq \mathrm{bwd}(G) + 1$$



$\rightsquigarrow$



# OUTLINE

- monoidal decompositions
- monoidal width for matrices
- monoidal width for path, tree & branch width
- monoidal width for rank width

# RANK WIDTH [Oum & Seymour, 2006]

$G = (V, E, \mathrm{ends} : E \to \mathcal{P}_{\leq 2}(V))$ undirected graph

## RANK DECOMPOSITION

$(Y, r)$ where

- $Y$ is a subcubic tree (= any node has at most 3 neighbours)
- $r$ : leaves $Y \xrightarrow{\simeq} V$ labelling bijection

## WIDTH OF $(Y, r)$

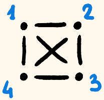
$$\mathrm{wd}(Y, r) := \max_{e \in \mathrm{edges}\, Y} \mathrm{rank}(X_e)$$

$X_e$ adjacency matrix of the cut given by e through $r$

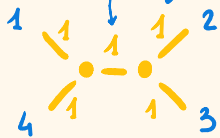
## RANK WIDTH

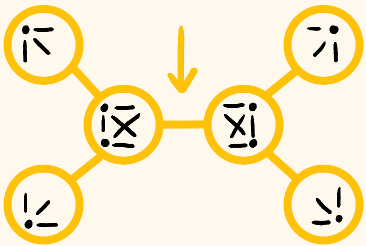
$$\mathrm{rwd}(G) := \min_{(Y, r)} \mathrm{wd}(Y, r)$$

# RANK WIDTH - EXAMPLE

G = 


$$\begin{array}{cc} & \begin{array}{cc} 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 4 \end{array} & \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cc} & \begin{array}{ccc} 1 & 3 & 4 \end{array} \\ 2 & \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \end{array}$$



# GRAPHS WITH DANGLING EDGES

$G = (V, E)$ undirected graph $\rightsquigarrow$ up to isomorphism

$\Rightarrow [G]$ with $G \in \mathrm{Mat}_{\mathbb{N}}(k,k)$ and

$$[G] = [H] \iff G + G^T = H + H^T$$

$\Gamma = ([G], B)$ graph with dangling edges $B \in \mathrm{Mat}_{\mathbb{N}}(k,n)$

$\Gamma =$  $= \left(\left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\right], \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}\right)$

$\rightsquigarrow$ graphs can be 'glued' along their dangling edges

# RANK DECOMPOSITIONS – RECURSIVELY

$\Gamma = ([G], B)$ graph with dangling edges

## RECURSIVE RANK DECOMPOSITION

$$T ::= (\Gamma) \qquad \text{if } \Gamma \text{ has at most one vertex}$$

$$\mid \quad \begin{tikzcd}[column sep=small] & \Gamma \arrow[dl, dash] \arrow[dr, dash] & \\ T_1 & & T_2 \end{tikzcd} \qquad \text{if } T_i \text{ rec. rank dec. of } \Gamma_i = ([G_i], B_i)$$

$$[G] = \left[\begin{pmatrix} G_1 & C \\ D & G_2 \end{pmatrix}\right] \; , \; B = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$$

$$B_1 = (A_1 \mid C) \; ; \; B_2 = (A_2 \mid C^T)$$

⇝ $\Gamma$ is obtained by 'glueing' $\Gamma_1$ and $\Gamma_2$

 =  'glued' with 

# RANK WIDTH - RECURSIVELY

$T$ recursive rank decomposition of $\Gamma = ([G], B)$

## WIDTH OF $T$

$$\mathrm{wd}(T) := \operatorname{rank} B \qquad \text{if } T = (\Gamma)$$

$$\mid \max\{\mathrm{wd}(T_1), \operatorname{rank} B, \mathrm{wd}(T_2)\} \qquad \text{if } T =$$

$$\begin{tikzcd}[column sep=small] & \Gamma \arrow[dl, dash] \arrow[dr, dash] & \\ T_1 & & T_2 \end{tikzcd}$$

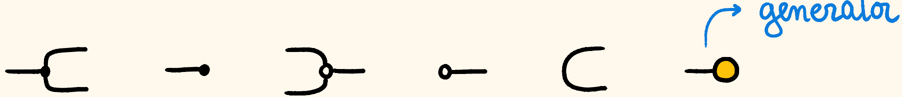
## RECURSIVE RANK WIDTH

$$\mathrm{rrwd}(\Gamma) := \min_{T} \mathrm{wd}(T)$$

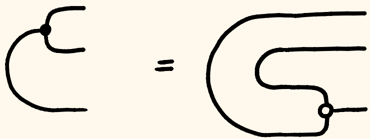
## PROPOSITION

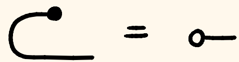
$$\mathrm{rwd}(G) \leq \mathrm{rrwd}(\Gamma) \leq \mathrm{rwd}(G) + \operatorname{rank} B$$

# A PROP OF GRAPHS

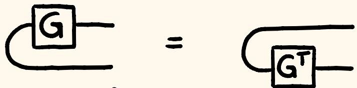



bialgebra equations +



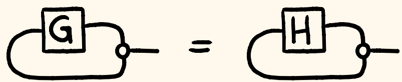


⇝ the cup transposes

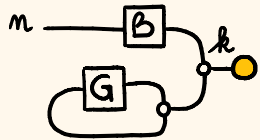


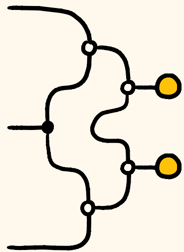

and captures equivalence of adjacency matrices

$[G] = [H] \iff$



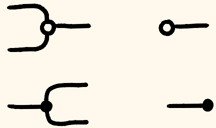
# A PROP OF GRAPHS – EXAMPLE

$\Gamma = ([G], B) \rightsquigarrow$ 


 $= \left(\left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\right], \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}\right) \rightsquigarrow$ 

# DECOMPOSITIONS IN THE PROP OF GRAPHS

Bialgebra structure



+ 'vertex' generator

## ATOMS

$\mathcal{A} = \{ \text{all morphisms} \}$

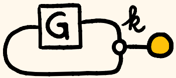
## WEIGHT FUNCTION

$w(g) := |\text{vertices } g|$

$w(i_n) := n$

# RANK WIDTH & MONOIDAL WIDTH

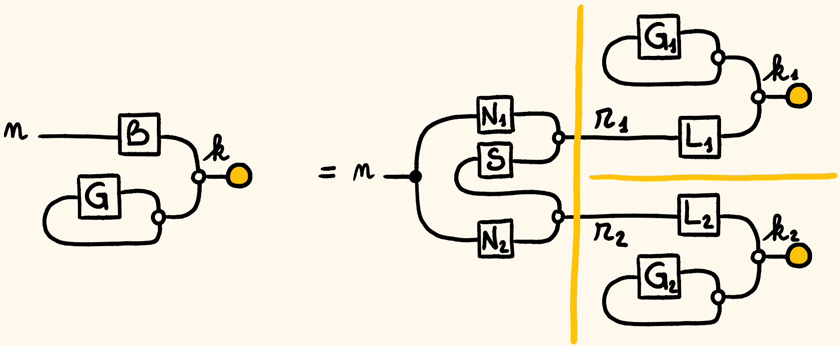
[G] undirected graph

$g =$ 

$: 0 \to 0$ in Grph

THEOREM

$$\frac{1}{2}\,\mathrm{rwd}(G) \leq \mathrm{mwd}(g) \leq 2\,\mathrm{rwd}(G)$$



# SUMMARY OF RESULTS

MATRICES

$$\max_i \operatorname{rank} A_i \leq \operatorname{mwd} A \leq \max_i \operatorname{rank} A_i + 1$$

COSPANS OF GRAPHS

$$\operatorname{pwd}(G) = \operatorname{mpwd}(g)$$

$$\operatorname{twd}(G) \leq \operatorname{mtwd}(g) \leq 2 \cdot \operatorname{twd}(G)$$

$$\frac{1}{2} \operatorname{bwd}(G) \leq \operatorname{mwd}(g) \leq \operatorname{bwd}(G) + 1$$

PROP OF GRAPHS

$$\frac{1}{2} \operatorname{rwd}(G) \leq \operatorname{mwd}(g) \leq 2 \operatorname{rwd}(G)$$

# FUTURE WORK

- obtain a result similar to Courcelle's theorem
- capture other widths (clique width, twin width, ... tree width for directed graphs and relational structures)
- algorithmic applications

# SOME REFERENCES


- Robertson & Seymour, Graph minors I-X, 1983-1991
- Oum & Seymour, Approximating clique width and branch width, 2006
- Arnborg, Courcelle, Proskurowski & Seese, An algebraic theory of graph reduction, 1993
- Courcelle & Olariu, Upper bounds to the clique width of graphs, 2000
- Berwanger, Dawar, Hunter, Kreutzer, Obdržálek, The DAG width of directed graphs, 2012
- Abramsky, Dawar & Pengming, The pebbling comonad in finite model theory, 2017
- Rosebrugh, Sabadini & Walters, Generic commutative separable algebras and cospans of graphs, 2005
- Chantawibul & Sobociński, Towards compositional graph theory, 2015
- Bonchi, Piedeleu, Sobociński & Zanasi, Graphical affine algebra, 2019
- Di Lavore, Hedges & Sobociński, Compositional modelling of network games, 2021


## THIS WORK


- Di Lavore & Sobociński, Monoidal width: unifying tree width, path width and branch width, 2022
- Di Lavore & Sobociński, Monoidal width: capturing rank width, 2022