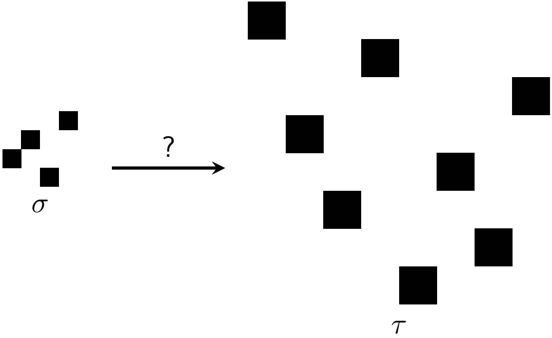
Introduction to twin-width

Édouard Bonnet

ENS Lyon, LIP

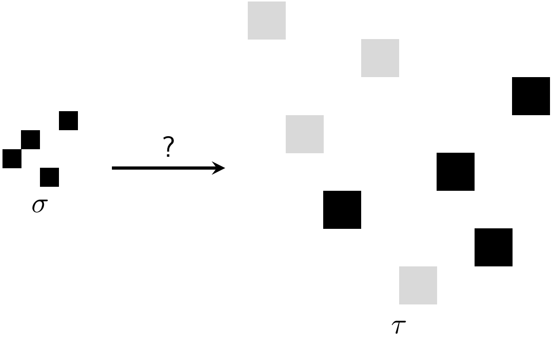
December 7th, Dresden Seminar
Algebra–Geometrie–Kombinatorik, Germany

The genesis: Permutation Pattern



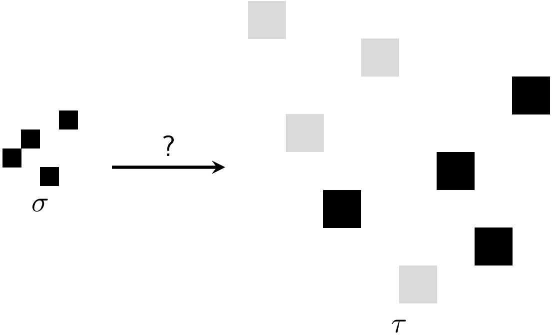
Is 3124 in 57362841?

The genesis: Permutation Pattern



Is 3124 in 5**736**2**8**41? Yes

The genesis: Permutation Pattern



Theorem (Guillemot, Marx '14)

Permutation Pattern *can be solved in time* $f(|\sigma|)|\tau|$.

Guillemot and Marx's win-win algorithm

Is σ in τ?

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ 1-entries has a t-grid minor.

4-grid minor

$$\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\hline
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

Guillemot and Marx's win-win algorithm

Is σ in τ?

Theorem (Marcus, Tardos '04)

$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ 1-entries has a t-grid minor.

4-grid minor

1	1	1	1	1	1	1	0
0	1	1	0	0	1	0	1
0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0
1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0
1	0	1	1	1	0	0	1

$\geqslant c_{|\sigma|} n$ 1-entries: answer YES from the $|\sigma|$-grid minor, or
$< c_{|\sigma|} n$ 1-entries: merge of two "similar" rectangles of 1s

Guillemot and Marx's win-win algorithm

Is σ in τ?

Theorem (Marcus, Tardos '04)

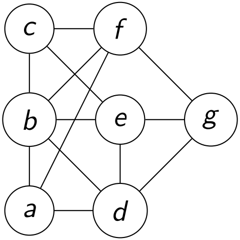
$\forall t$, $\exists c_t$ $\forall$ $n \times n$ 0,1-matrix with $\geqslant c_t n$ 1-entries has a t-grid minor.

4-grid minor
$$\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\right]$$

$\geqslant c_{|\sigma|} n$ 1-entries: answer YES from the $|\sigma|$-grid minor, or
$< c_{|\sigma|} n$ 1-entries: merge of two "similar" rectangles of 1s

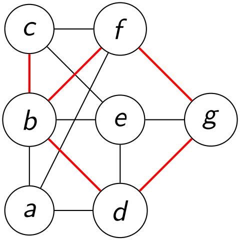
If the latter always holds: exploitable "decomposition" of τ

Graphs



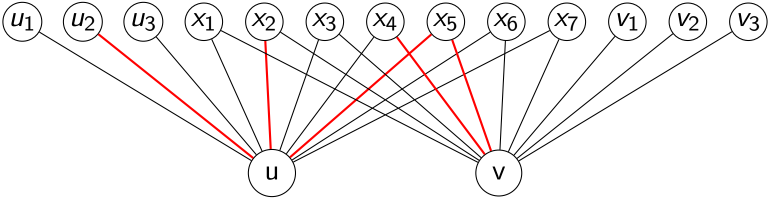
Two outcomes between a pair of vertices:
edge or non-edge

Trigraphs



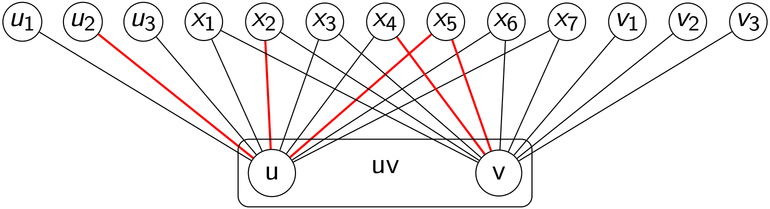
Three outcomes between a pair of vertices:
edge, or non-edge, or red edge (error edge)

Contractions in trigraphs



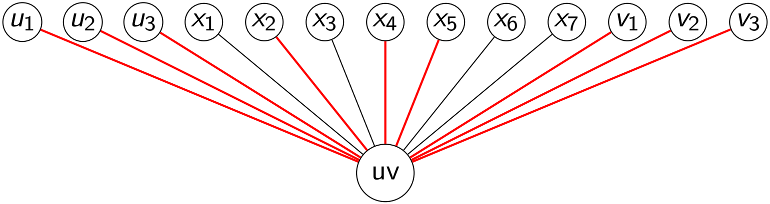
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



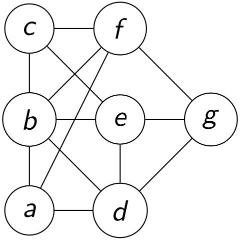
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



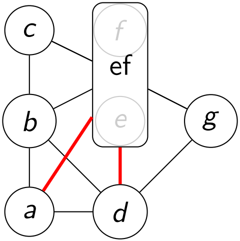
edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing

Contraction sequence



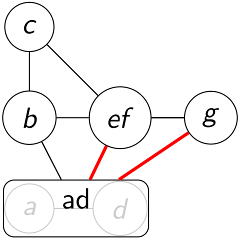
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \dots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



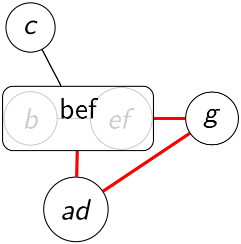
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



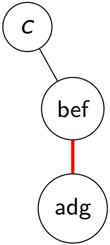
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



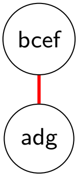
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



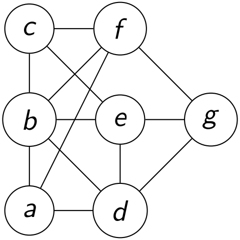
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence

A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \dots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

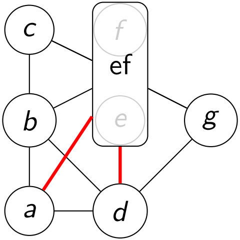


Maximum red degree = 0

overall maximum red degree = 0

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

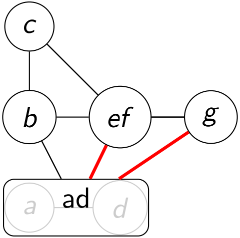


Maximum red degree = 2

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

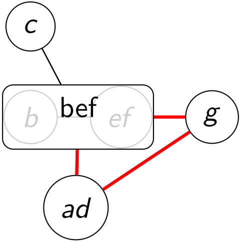


Maximum red degree = 2

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

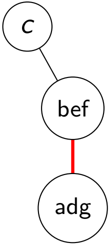


Maximum red degree = 2

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

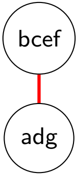


Maximum red degree $= 1$

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.



Maximum red degree $= 1$

overall maximum red degree = 2

Twin-width

tww(G): Least integer d such that G admits a contraction sequence where all trigraphs have *maximum red degree* at most d.

Maximum red degree $= 0$

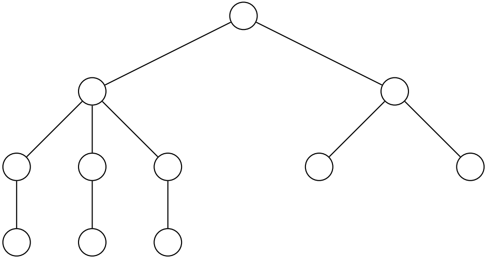
overall maximum red degree = 2

Extension to binary structures over a finite signature

- Red edges appear between two vertices X, Y such that, for some binary relation R, $R(x, y)$ holds for some $x \in X$ and $y \in Y$, and $R(x', y')$ does not, for some $x' \in X$ and $y' \in Y$.
- Contraction only allowed within vertices satisfying the same unary relations.

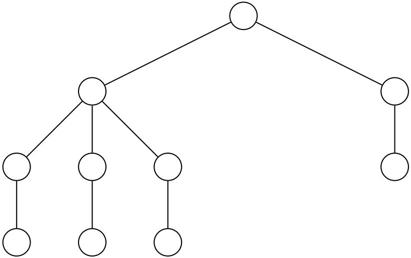
We now contract to up to 2^h remaining vertices, with h the number of unary relations.

Trees



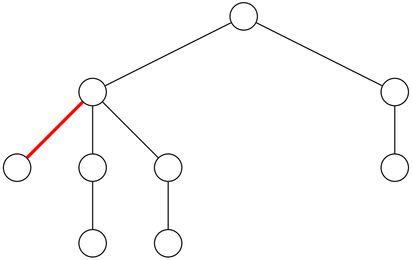
If possible, contract two twin leaves

Trees



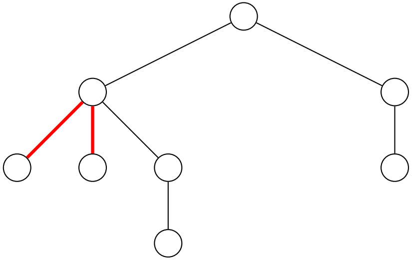
If not, contract a deepest leaf with its parent

Trees



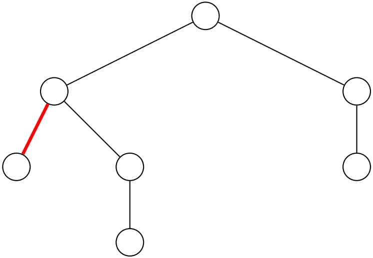
If not, contract a deepest leaf with its parent

Trees



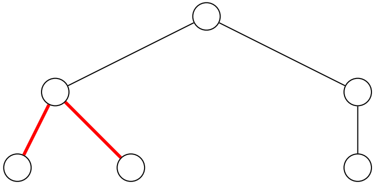
If possible, contract two twin leaves

Trees



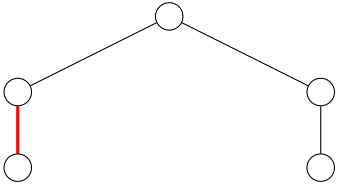
Cannot create a red degree-3 vertex

Trees



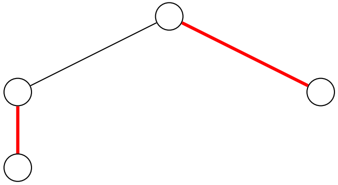
Cannot create a red degree-3 vertex

Trees



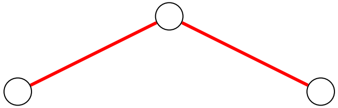
Cannot create a red degree-3 vertex

Trees



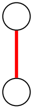
Cannot create a red degree-3 vertex

Trees



Cannot create a red degree-3 vertex

Trees

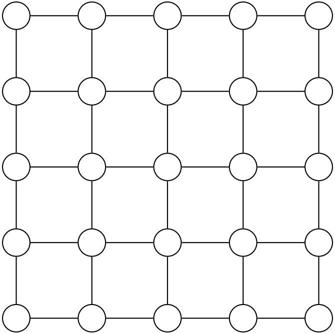


Cannot create a red degree-3 vertex

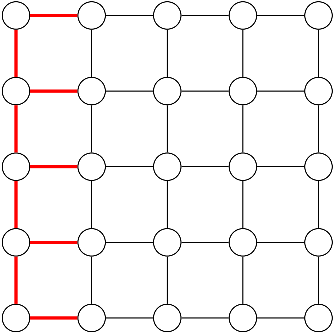
Trees

Generalization to bounded *treewidth* and even bounded *rank-width*

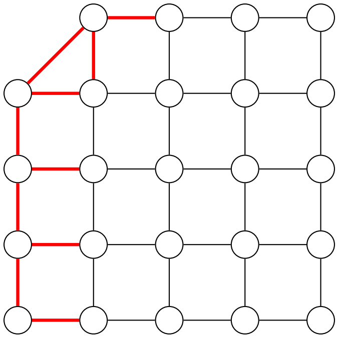
Grids



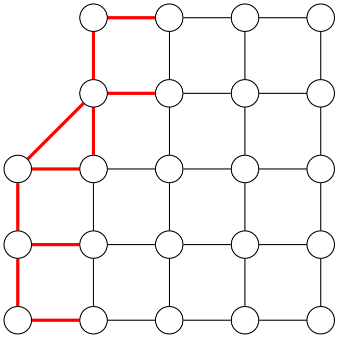
Grids



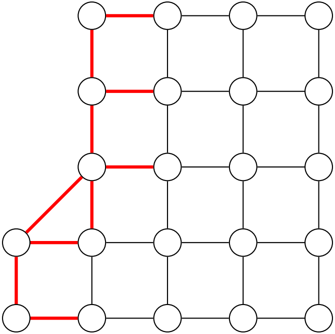
Grids



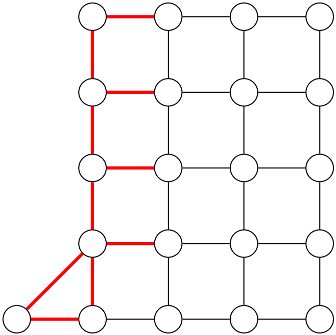
Grids



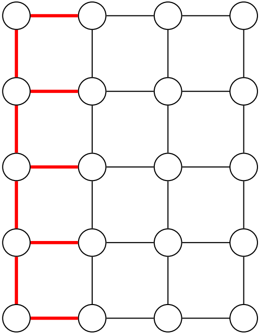
Grids



Grids



Grids



4-sequence for planar grids

Marcus–Tardos-like characterization of bounded twin-width

Mixed cell = not horizontal nor vertical

$$\begin{bmatrix}\begin{array}{cc|ccc|ccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}\end{bmatrix}$$

3-mixed minor

k-mixed minor = k-division where every cell is mixed

Marcus–Tardos-like characterization of bounded twin-width

Mixed cell = not horizontal nor vertical

$$\left[\begin{array}{cc|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

3-mixed minor

k-mixed minor = k-division where every cell is mixed

Mixed number of a graph G =
$\min_{<} \max\{k : \text{Adj}_<(G) \text{ has a } k\text{-mixed minor}\}$

Theorem (B., Kim, Thomassé, Watrigant '20)

A class has bounded twin-width iff it has bounded mixed number.

Marcus–Tardos-like characterization of bounded twin-width

Mixed cell = not horizontal nor vertical

$$\left[\begin{array}{cc|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

3-mixed minor

k-mixed minor = k-division where every cell is mixed

Grid rank of a graph $G =$
$\min_{<} \max\{k : \mathrm{Adj}_<(G) \text{ has a } k\text{-division with all cells of rank} \geqslant k\}$

Theorem (B., Giocanti, Ossona de Mendez, Simon, Thomassé, Toruńczyk '22)

A class has bounded twin-width iff it has bounded grid rank.

Theorem (B., Geniet, Kim, Thomassé, Watrigant '20 & '21)

The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

- *Bounded rank-width, and even, boolean-width graphs,*
- *every hereditary proper subclass of permutation graphs,*
- *posets of bounded antichain size (seen as digraphs),*
- *unit interval graphs,*
- *K_t-minor free graphs,*
- *map graphs,*
- *subgraphs of d-dimensional grids,*
- *K_t-free unit d-dimensional ball graphs,*
- *$\Omega(\log n)$-subdivisions of all the n-vertex graphs,*
- *cubic expanders defined by iterative random 2-lifts from K_4,*
- *strong products of two bounded twin-width classes, one with bounded degree, etc.*

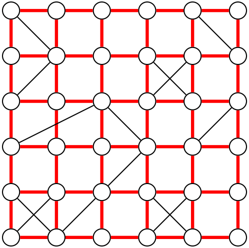
Theorem (B., Geniet, Kim, Thomassé, Watrigant '20 & '21)

The following classes have bounded twin-width, and $O(1)$-sequences can be computed in polynomial time.

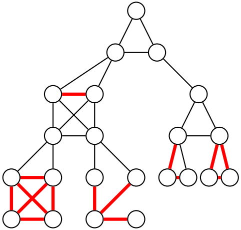
- *Bounded rank-width, and even, boolean-width graphs,*
- *every hereditary proper subclass of permutation graphs,*
- *posets of bounded antichain size (seen as digraphs),*
- *unit interval graphs,*
- *K_t-minor free graphs,*
- *map graphs,*
- *subgraphs of d-dimensional grids,*
- *K_t-free unit d-dimensional ball graphs,*
- *$\Omega(\log n)$-subdivisions of all the n-vertex graphs,*
- *cubic expanders defined by iterative random 2-lifts from K_4,*
- *strong products of two bounded twin-width classes, one with bounded degree, etc.*

Ok, but do bounded twin-width classes have good properties?

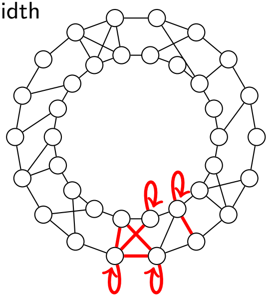
Different conditions imposed in the sequence of red graphs



bd degree: defines bd twin-width



bd component: redefines bd cliquewidth



bd #edges: redefines bd linear cliquewidth

Graph model checking

Graph FO/MSO Model Checking **Parameter:** $|\varphi|$

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$

Question: $G \models \varphi$?

Graph model checking

> Graph FO/MSO Model Checking **Parameter:** $|\varphi|$
> **Input:** A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
> **Question:** $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leqslant i \leqslant k} x = x_i \vee \bigvee_{1 \leqslant i \leqslant k} E(x, x_i) \vee E(x_i, x)$$

$G \models \varphi$? $\Leftrightarrow$

Graph model checking

GRAPH FO/MSO MODEL CHECKING **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \forall x \bigvee_{1 \leqslant i \leqslant k} x = x_i \vee \bigvee_{1 \leqslant i \leqslant k} E(x, x_i) \vee E(x_i, x)$$

$G \models \varphi$? $\Leftrightarrow$ k-DOMINATING SET

Graph model checking

GRAPH FO/MSO MODEL CHECKING	**Parameter:** $\|\varphi\|$

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \bigwedge_{1 \leqslant i < j \leqslant k} \neg(x_i = x_j) \wedge \neg E(x_i, x_j) \wedge \neg E(x_j, x_i)$$

$G \models \varphi$? $\Leftrightarrow$

Graph model checking

GRAPH FO/MSO MODEL CHECKING **Parameter:** $|\varphi|$
Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists x_1 \exists x_2 \cdots \exists x_k \bigwedge_{1 \leqslant i < j \leqslant k} \neg(x_i = x_j) \wedge \neg E(x_i, x_j) \wedge \neg E(x_j, x_i)$$

$G \models \varphi$? $\Leftrightarrow$ k-INDEPENDENT SET

Graph model checking

GRAPH FO/MSO MODEL CHECKING	**Parameter:** $\|\varphi\|$

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists X_1 \exists X_2 \exists X_3 (\forall x \bigvee_{1 \leqslant i \leqslant 3} X_i(x)) \wedge \forall x \forall y \bigwedge_{1 \leqslant i \leqslant 3} (X_i(x) \wedge X_i(y) \to \neg E(x, y))$$

$G \models \varphi? \Leftrightarrow$

Graph model checking

GRAPH FO/MSO MODEL CHECKING	**Parameter:** $\lvert\varphi\rvert$

Input: A graph G and a first-order/monadic second-order sentence $\varphi \in FO/MSO(\{E\})$
Question: $G \models \varphi$?

Example:

$$\varphi = \exists X_1 \exists X_2 \exists X_3 (\forall x \bigvee_{1 \leqslant i \leqslant 3} X_i(x)) \wedge \forall x \forall y \bigwedge_{1 \leqslant i \leqslant 3} (X_i(x) \wedge X_i(y) \rightarrow \neg E(x, y))$$

$G \models \varphi$? $\Leftrightarrow$ 3-COLORING

The lens of contraction sequences

Class of bounded	constraint on red graphs	efficient model-checking
linear rank-width	bd #edges	MSO
rank-width	bd component	**MSO**
twin-width	bd degree	**?**

The lens of contraction sequences

Class of bounded	constraint on red graphs	efficient model-checking
linear rank-width	bd #edges	MSO
rank-width	bd component	**MSO**
twin-width	bd degree	**?**

We will reprove the result in bold, and fill the **?**

Courcelle's theorems

We will reprove with contraction sequences:

Theorem (Courcelle, Makowsky, Rotics '00)

MSO model checking can be solved in time $f(|\varphi|, d) \cdot |V(G)|$ *given a witness that the clique-width/component twin-width of the input* G *is at most* d.

generalizes

Theorem (Courcelle '90)

MSO model checking can be solved in time $f(|\varphi|, t) \cdot |V(G)|$ *on incidence graphs of graphs* G *of treewidth at most* t.

Rank-k m-types

Sets of non-equivalent formulas/sentences of quantifier rank at most k satisfied by a fixed structure:

$$\mathrm{tp}_k^{\mathcal{L}}(\mathscr{A}, \vec{a} \in A^m) = \{\varphi(\vec{x}) \in \mathcal{L}[k] : \mathscr{A} \models \varphi(\vec{a})\},$$

$$\mathrm{tp}_k^{\mathcal{L}}(\mathscr{A}) = \{\varphi \in \mathcal{L}[k] : \mathscr{A} \models \varphi\}.$$

Rank-k m-types

Sets of non-equivalent formulas/sentences of quantifier rank at most k satisfied by a fixed structure:

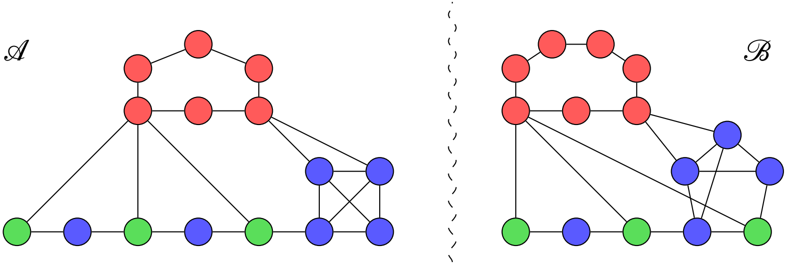
$$\mathrm{tp}_k^{\mathcal{L}}(\mathscr{A}, \vec{a} \in A^m) = \{\varphi(\vec{x}) \in \mathcal{L}[k] : \mathscr{A} \models \varphi(\vec{a})\},$$

$$\mathrm{tp}_k^{\mathcal{L}}(\mathscr{A}) = \{\varphi \in \mathcal{L}[k] : \mathscr{A} \models \varphi\}.$$

Fact

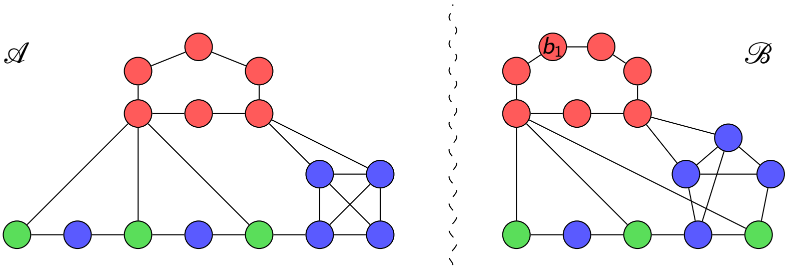
For $\mathcal{L} \in \{FO, MSO\}$, the number of rank-$k$ m-types is bounded by a function of k and m only.

FO Ehrenfeucht-Fraissé game



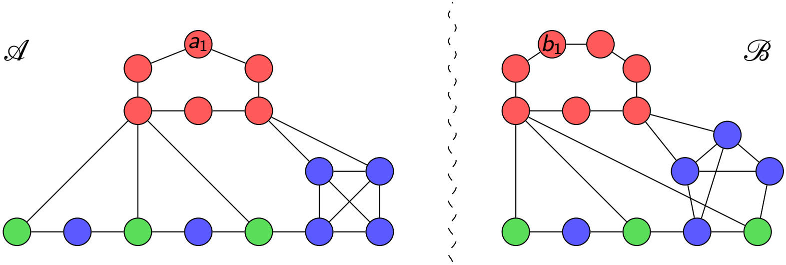
2-player game on two σ-structures $\mathscr{A}, \mathscr{B}$ (for us, colored graphs)

FO Ehrenfeucht-Fraissé game



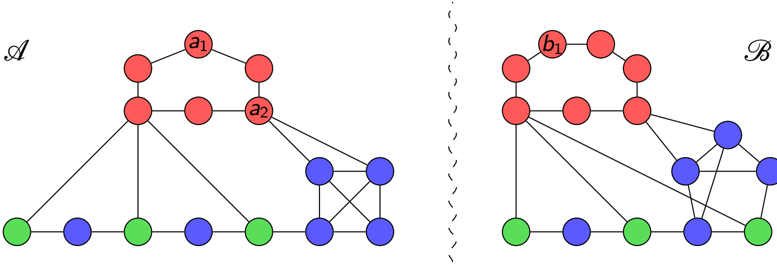
At each round, Spoiler picks a structure ($\mathscr{B}$) and a vertex therein

FO Ehrenfeucht-Fraissé game



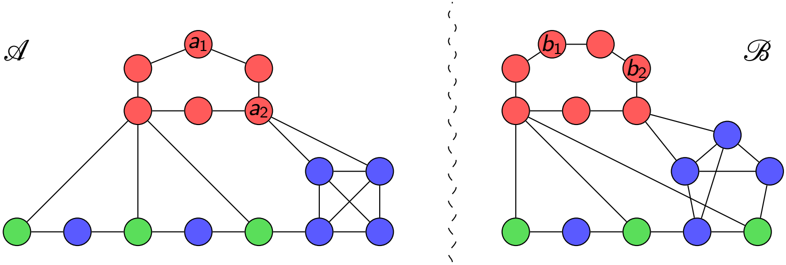
Duplicator answers with a vertex in the other structure

FO Ehrenfeucht-Fraissé game



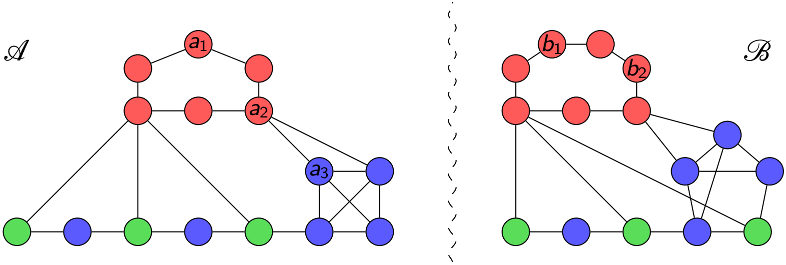
After q rounds, Duplicator wishes that $a_i \mapsto b_i$ is an isomorphism between $\mathscr{A}[a_1, \ldots, a_k]$ and $\mathscr{B}[b_1, \ldots, b_k]$

FO Ehrenfeucht-Fraissé game



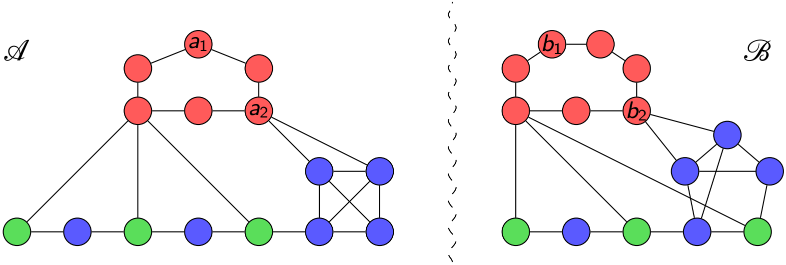
After q rounds, Duplicator wishes that $a_i \mapsto b_i$ is an isomorphism between $\mathscr{A}[a_1, \ldots, a_k]$ and $\mathscr{B}[b_1, \ldots, b_k]$

FO Ehrenfeucht-Fraissé game



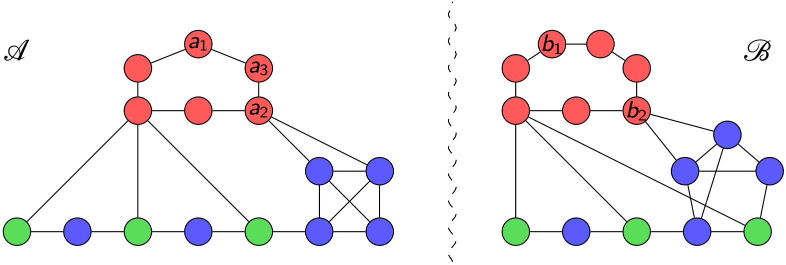
When no longer possible, Spoiler wins

FO Ehrenfeucht-Fraissé game



When no longer possible, Spoiler wins

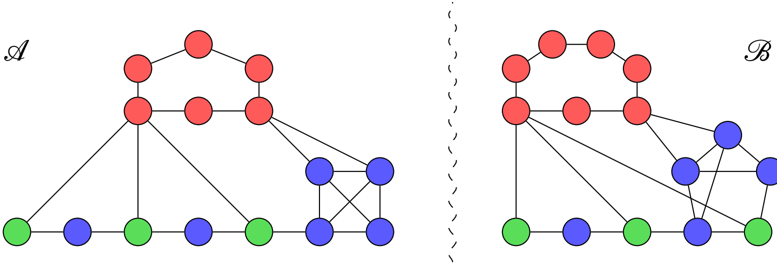
FO Ehrenfeucht-Fraissé game



If Duplicator can survive k rounds, we write $\mathscr{A} \equiv_k^{\mathsf{FO}} \mathscr{B}$

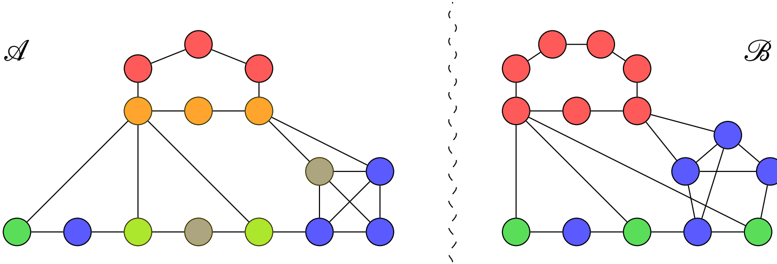
Here $\mathscr{A} \equiv_2^{\mathsf{FO}} \mathscr{B}$ and $\mathscr{A} \not\equiv_3^{\mathsf{FO}} \mathscr{B}$

MSO Ehrenfeucht-Fraissé game



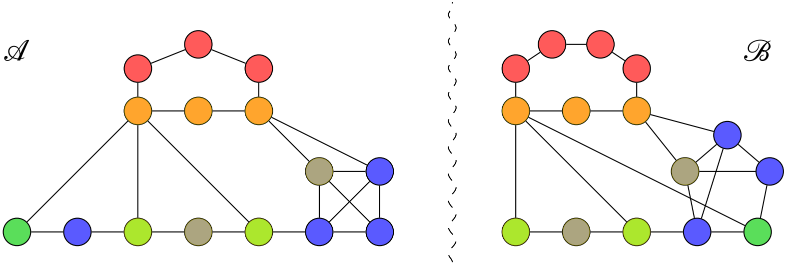
Same game but Spoiler can now play set moves

MSO Ehrenfeucht-Fraissé game



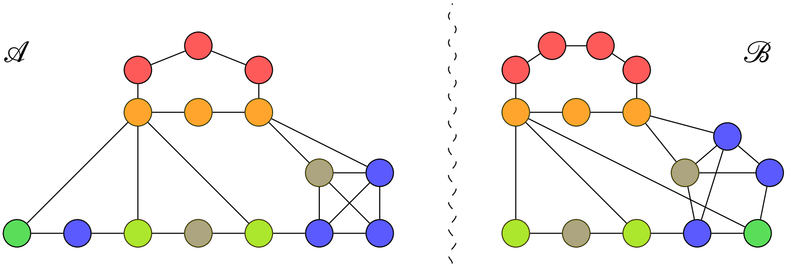
Same game but Spoiler can now play set moves

MSO Ehrenfeucht-Fraissé game



To which Duplicator answers a set in the other structure

MSO Ehrenfeucht-Fraissé game



Again we write $\mathscr{A} \equiv_k^{\mathsf{MSO}} \mathscr{B}$ if Duplicator can survive k rounds

k-round EF games capture rank-k types

Theorem (Ehrenfeucht-Fraïssé)

For every σ-structures $\mathscr{A}, \mathscr{B}$ and logic $\mathcal{L} \in \{FO, MSO\}$,

$$\mathscr{A} \equiv_k^{\mathcal{L}} \mathscr{B} \text{ if and only if } tp_k^{\mathcal{L}}(\mathscr{A}) = tp_k^{\mathcal{L}}(\mathscr{B}).$$

k-round EF games capture rank-k types

Theorem (Ehrenfeucht-Fraissé)

For every σ-structures $\mathscr{A}, \mathscr{B}$ and logic $\mathcal{L} \in \{FO, MSO\}$,

$$\mathscr{A} \equiv_k^{\mathcal{L}} \mathscr{B} \text{ if and only if } tp_k^{\mathcal{L}}(\mathscr{A}) = tp_k^{\mathcal{L}}(\mathscr{B}).$$

Proof.

Induction on k.

($\Rightarrow$) $\mathcal{L}[k+1]$ formulas are Boolean combinations of $\exists x \varphi$ or $\exists X \varphi$ where $\varphi \in \mathcal{L}[k]$. Use the answer of Duplicator to $x = a$ or $X = A$.

k-round EF games capture rank-k types

Theorem (Ehrenfeucht-Fraissé)

For every σ-structures $\mathscr{A}, \mathscr{B}$ and logic $\mathcal{L} \in \{FO, MSO\}$,

$$\mathscr{A} \equiv_k^{\mathcal{L}} \mathscr{B} \text{ if and only if } tp_k^{\mathcal{L}}(\mathscr{A}) = tp_k^{\mathcal{L}}(\mathscr{B}).$$

Proof.

Induction on k.

($\Rightarrow$) $\mathcal{L}[k+1]$ formulas are Boolean combinations of $\exists x \varphi$ or $\exists X \varphi$ where $\varphi \in \mathcal{L}[k]$. Use the answer of Duplicator to $x = a$ or $X = A$.

($\Leftarrow$) If $\mathrm{tp}_{k+1}^{\mathcal{L}}(\mathcal{A}) = \mathrm{tp}_{k+1}^{\mathcal{L}}(\mathcal{B})$, then the type $\mathrm{tp}_k^{\mathcal{L}}(\mathcal{A}, a)$ is equal to some $\mathrm{tp}_k^{\mathcal{L}}(\mathcal{B}, b)$. Move a can be answered by playing b. □

MSO model checking for component twin-width d

Partitioned sentences: sentences on $(E, U_1, \ldots, U_d)$-structures, interpreted as a graph vertex partitioned in d parts

Maintain for every red component C of every trigraph G_i

$$\mathrm{tp}_k^{\mathrm{MSO}}(G, \mathcal{P}_i, C) = \{\varphi \in \mathrm{MSO}_{E,U_1,\ldots,U_d}[k] : (G\langle C\rangle, \mathcal{P}_i\langle C\rangle) \models \varphi\}.$$

MSO model checking for component twin-width d

Partitioned sentences: sentences on $(E, U_1, \ldots, U_d)$-structures, interpreted as a graph vertex partitioned in d parts

Maintain for every red component C of every trigraph G_i

$$\text{tp}_k^{\text{MSO}}(G, \mathcal{P}_i, C) = \{\varphi \in \text{MSO}_{E,U_1,\ldots,U_d}[k] : (G\langle C\rangle, \mathcal{P}_i\langle C\rangle) \models \varphi\}.$$

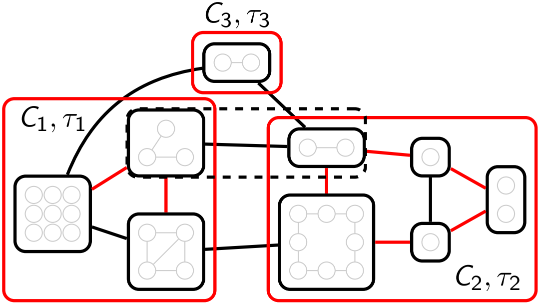
For each $v \in V(G)$, $\text{tp}_k(G, \mathcal{P}_n, \{v\}) =$ type of K_1
$\text{tp}_k(G, \mathcal{P}_1, \{V(G)\}) =$ type of G

MSO model checking for component twin-width d

Partitioned sentences: sentences on $(E, U_1, \ldots, U_d)$-structures, interpreted as a graph vertex partitioned in d parts

Maintain for every red component C of every trigraph G_i

$$\text{tp}_k^{\text{MSO}}(G, \mathcal{P}_i, C) = \{\varphi \in \text{MSO}_{E,U_1,\ldots,U_d}[k] : (G\langle C\rangle, \mathcal{P}_i\langle C\rangle) \models \varphi\}.$$



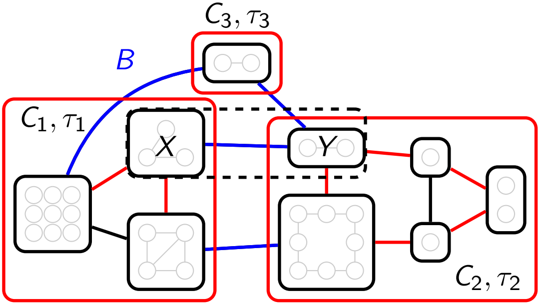
$\tau = \text{tp}_k^{\text{MSO}}(G, \mathcal{P}_i, C)$ based on the $\tau_j = \text{tp}_k^{\text{MSO}}(G, \mathcal{P}_{i+1}, C_j)$?

MSO model checking for component twin-width d

Partitioned sentences: sentences on $(E, U_1, \ldots, U_d)$-structures, interpreted as a graph vertex partitioned in d parts

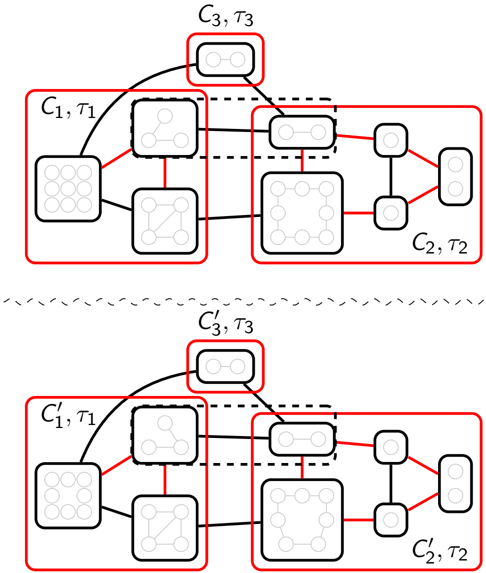
Maintain for every red component C of every trigraph G_i

$$\mathrm{tp}_k^{\mathrm{MSO}}(G, \mathcal{P}_i, C) = \{\varphi \in \mathrm{MSO}_{E,U_1,\ldots,U_d}[k] : (G\langle C \rangle, \mathcal{P}_i\langle C \rangle) \models \varphi\}.$$



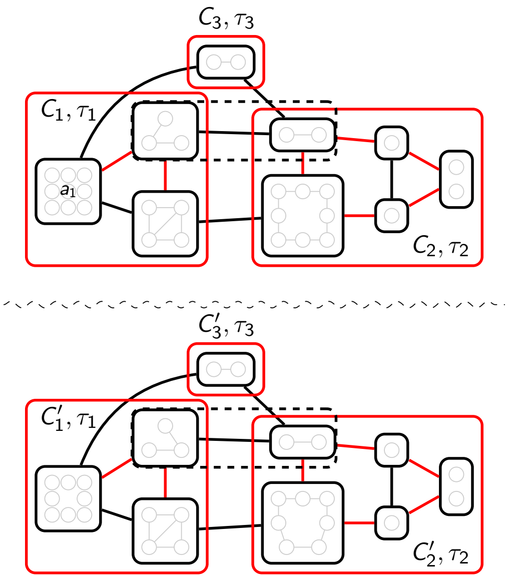
C arises from $C_1, \ldots, C_{d'}$: $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



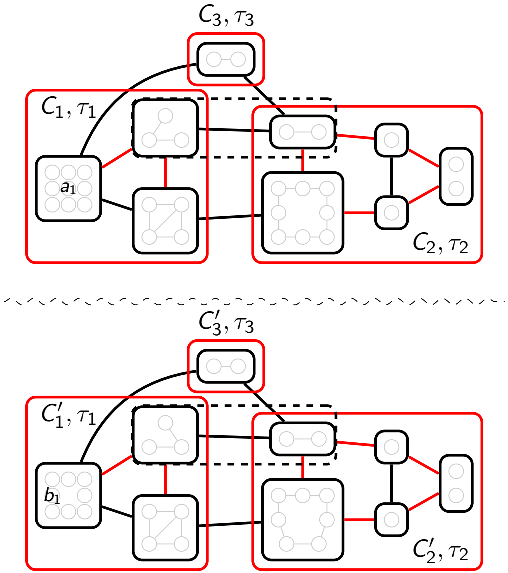
Duplicator combines her strategies in the red components

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



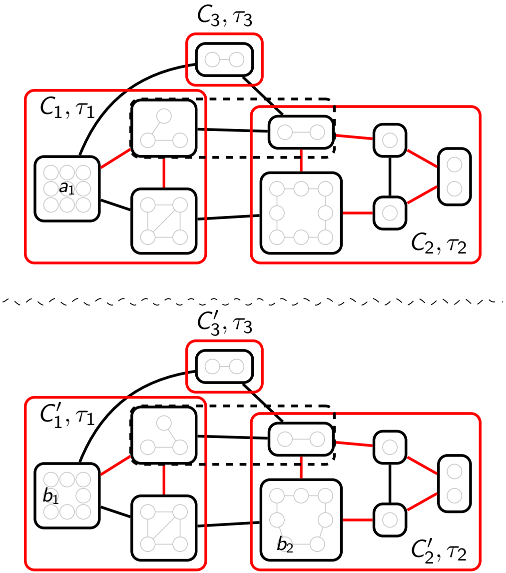
If Spoiler plays a vertex in the component of type τ_1,

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



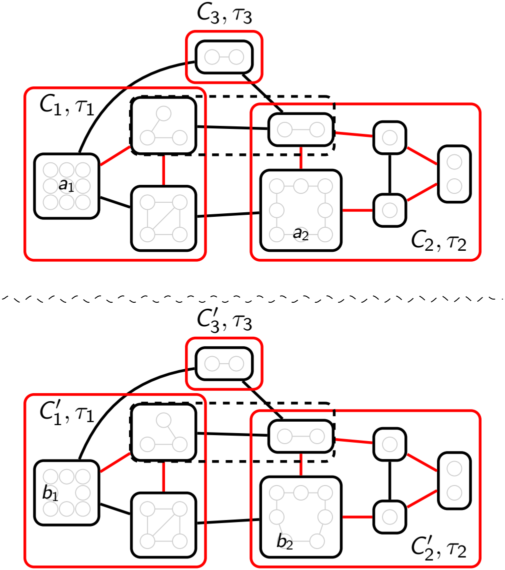
Duplicator answers the corresponding winning move

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



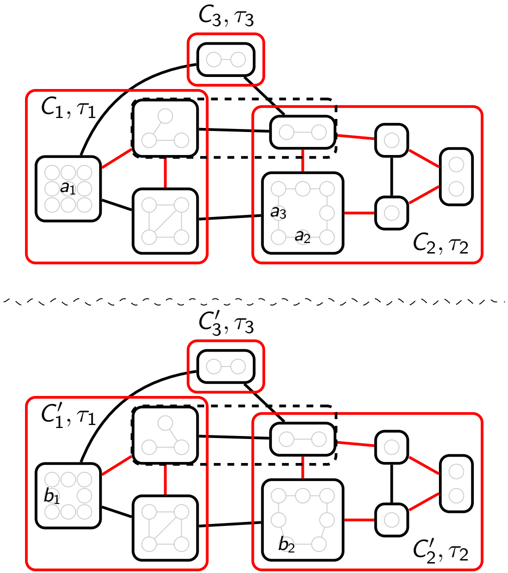
Same in the component of type τ_2

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



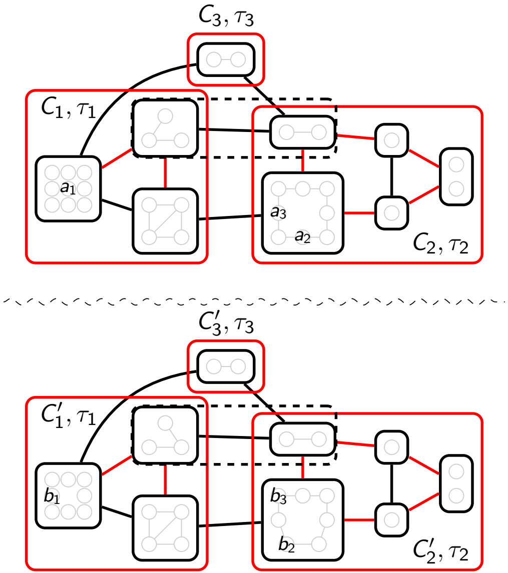
Same in the component of type τ_2

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



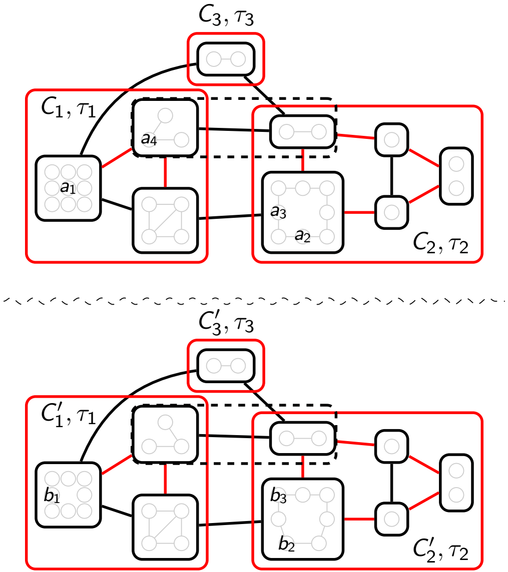
Same in the component of type τ_2

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



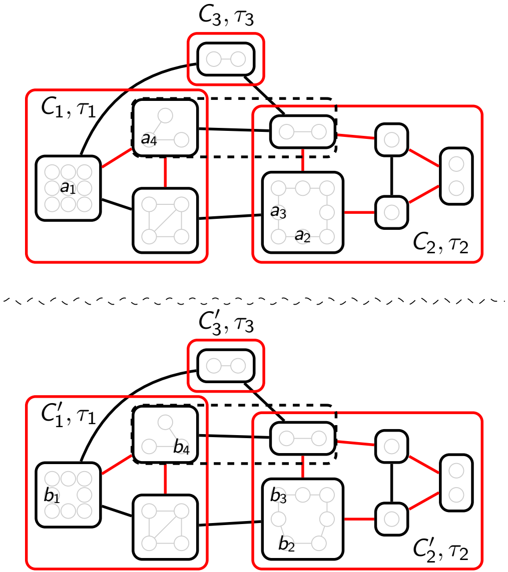
Same in the component of type τ_2

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



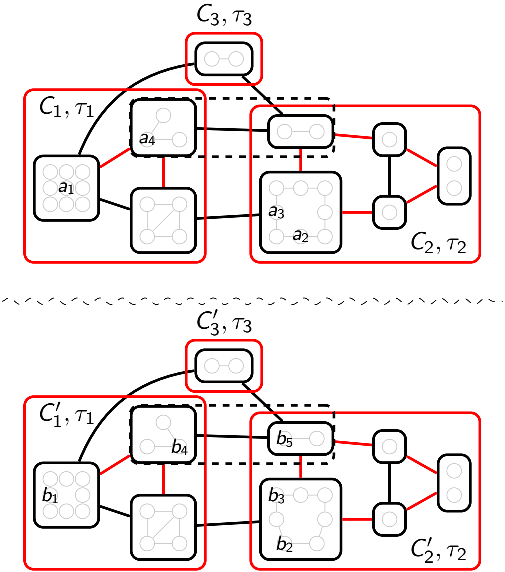
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



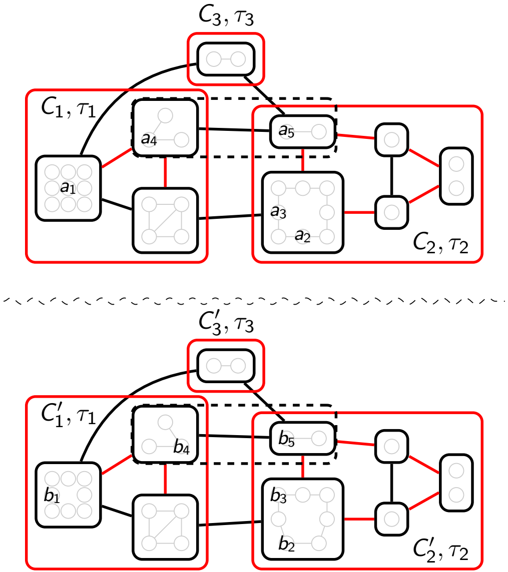
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



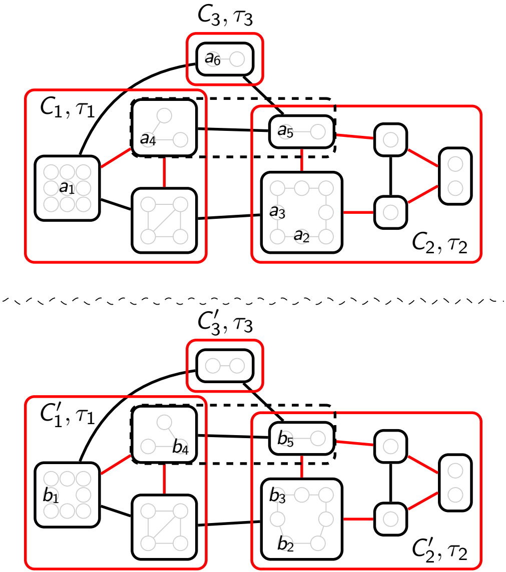
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



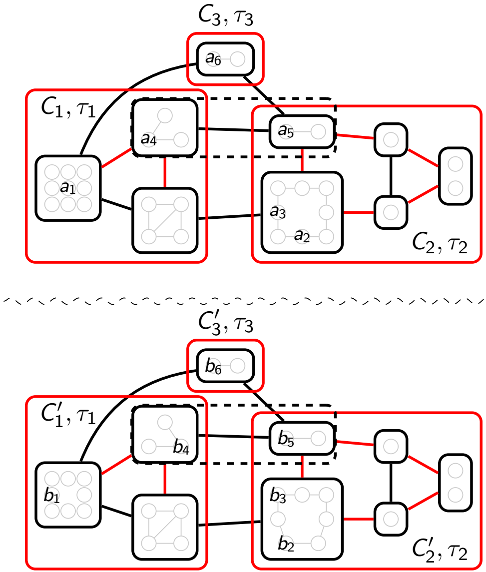
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



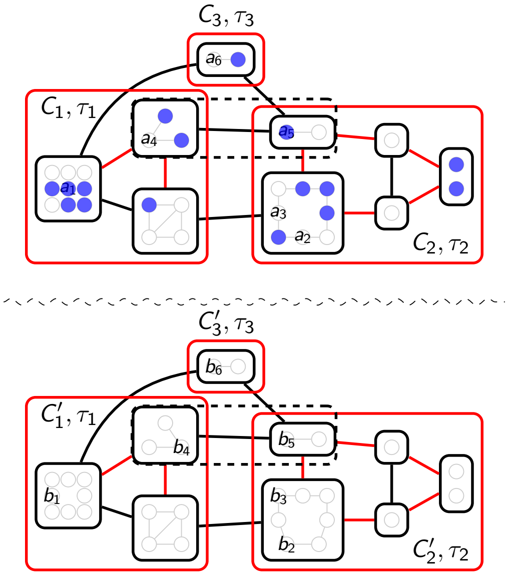
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



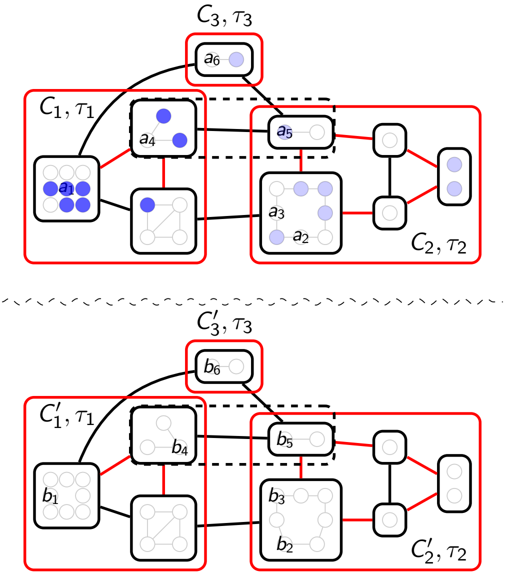
and so on

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



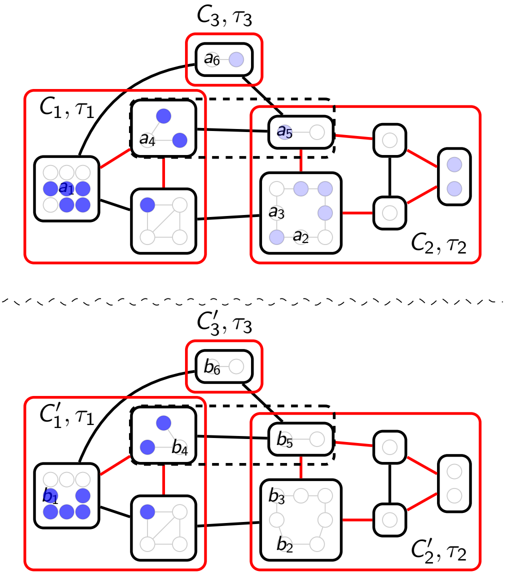
If Spoiler plays a set, Duplicator looks at the intersection with C_1,

Showing $\tau = F(\tau_1, \dots, \tau_{d'}, B, X, Y)$ via MSO EF game



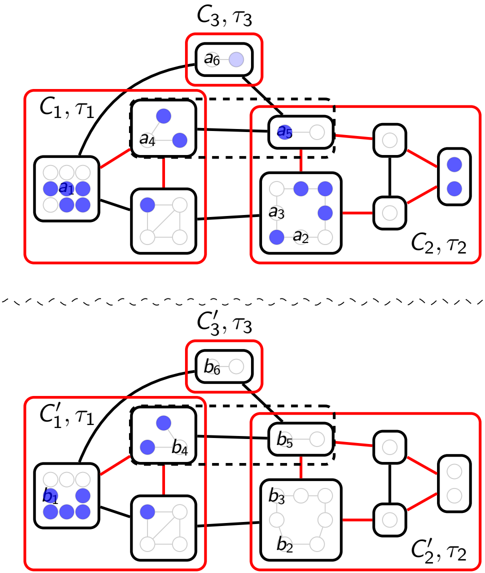
If Spoiler plays a set, Duplicator looks at the intersection with C_1,

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



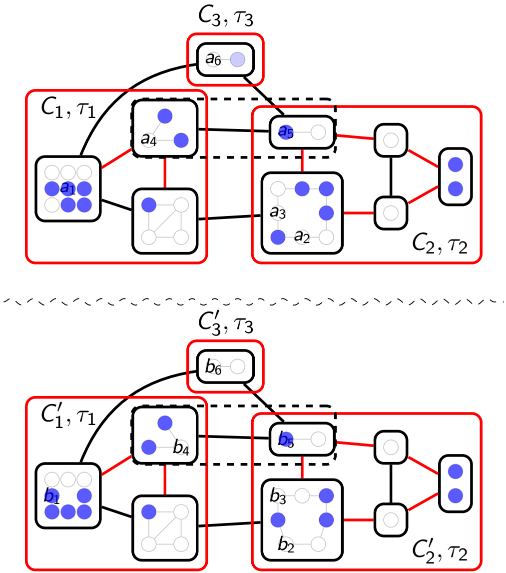
calls her winning strategy in C'_1

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



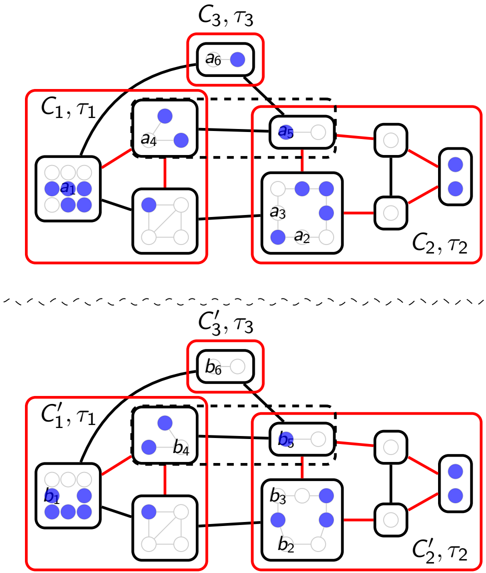
same for the other components

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



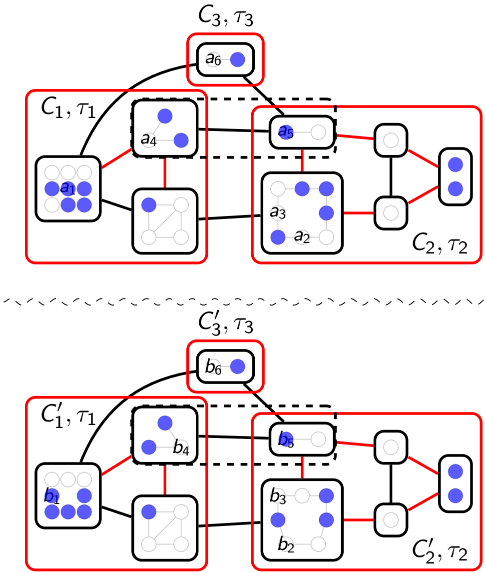
same for the other components

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



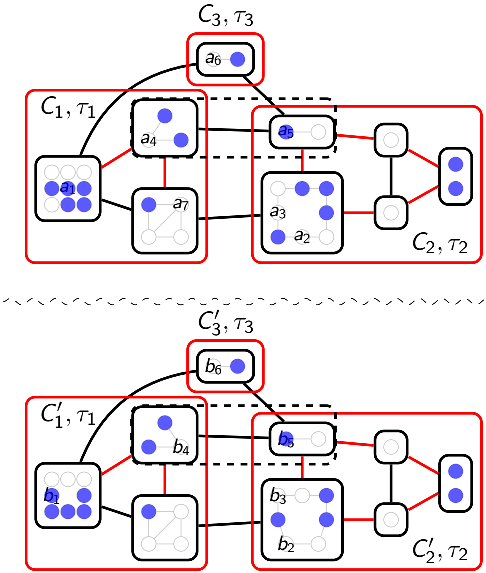
same for the other components

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



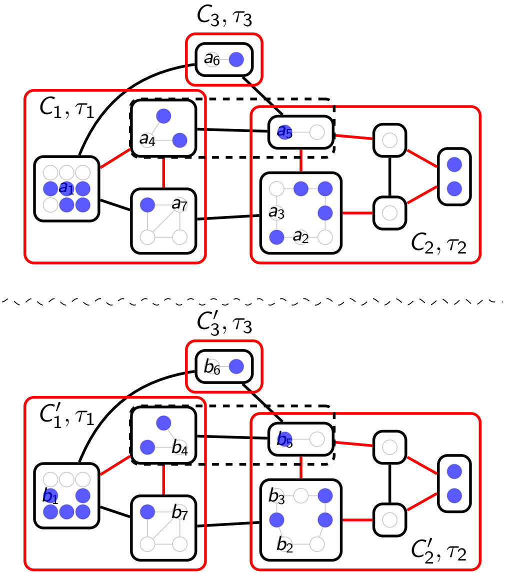
and plays the union

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



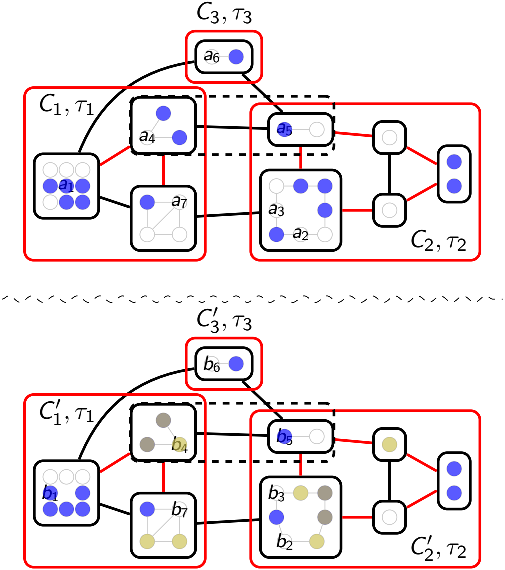
that fully defines the winning strategy of Duplicator

Showing $\tau = F(\tau_1, \dots, \tau_{d'}, B, X, Y)$ via MSO EF game



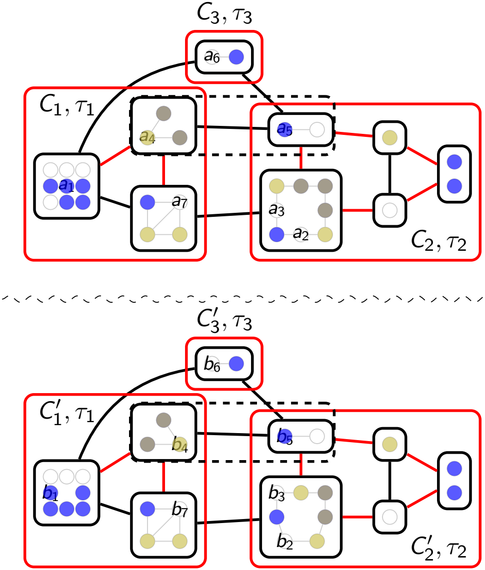
that fully defines the winning strategy of Duplicator

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



that fully defines the winning strategy of Duplicator

Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



that fully defines the winning strategy of Duplicator

Turning it into a uniform algorithm

Reminder:

- #non-equivalent partitioned sentences of rank k: $f(d,k)$
- #rank-k partitioned types bounded by $g(d,k) = 2^{f(d,k)}$

For each newly observed type τ,

- keep a representative $(H,\mathcal{P})_\tau$ on at most $(d+1)^{g(d,k)}$ vertices
- determine the $0,1$-vector of satisfied sentences on $(H,\mathcal{P})_\tau$
- record the value of $F(\tau_1,\ldots,\tau_{d'},B,X,Y)$ for future uses

Turning it into a uniform algorithm

Reminder:

- #non-equivalent partitioned sentences of rank k: $f(d,k)$
- #rank-k partitioned types bounded by $g(d,k) = 2^{f(d,k)}$

For each newly observed type τ,

- keep a representative $(H,\mathcal{P})_\tau$ on at most $(d+1)^{g(d,k)}$ vertices
- determine the $0,1$-vector of satisfied sentences on $(H,\mathcal{P})_\tau$
- record the value of $F(\tau_1,\ldots,\tau_{d'},B,X,Y)$ for future uses

To decide $G \models \varphi$, look at position φ in the $0,1$-vector of $\mathrm{tp}_k^{\mathsf{MSO}}(G)$

Back to twin-width

k-Independent Set given a d-sequence

Complexity theory says that algorithms in time $f(k)|V(G)|^{o(k)}$ are unlikely to exist in general graphs

$d^k|V(G)|$ is possible with a d-sequence $G = G_n, \ldots, G_1$

Algorithm: **For every** $D \in \binom{V(G_i)}{\leqslant k}$ **such that** $\mathcal{R}(G_i)[D]$ **is connected, store in** $T[D, i]$ **one largest independent set in** $G\langle D \rangle$ **intersecting every vertex of** D**.**

k-Independent Set given a d-sequence

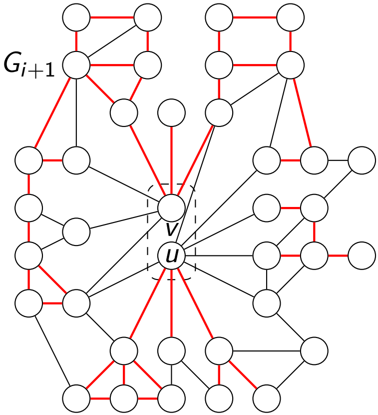
Complexity theory says that algorithms in time $f(k)|V(G)|^{o(k)}$ are unlikely to exist in general graphs

$d^k|V(G)|$ is possible with a d-sequence $G = G_n, \ldots, G_1$

Algorithm: **For every $D \in \binom{V(G_i)}{\leqslant k}$ such that $\mathcal{R}(G_i)[D]$ is connected, store in $T[D, i]$ one largest independent set in $G\langle D \rangle$ intersecting every vertex of D.**

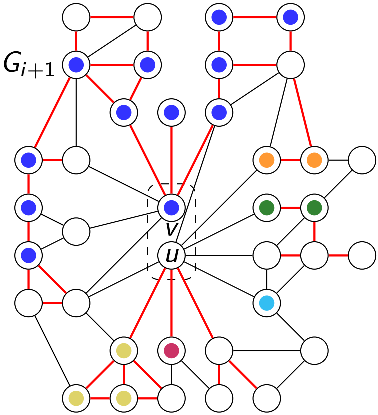
How to compute $T[D, i]$ from all the $T[D', i+1]$?

k-Independent Set: Update of partial solutions



Best partial solution inhabiting •?

k-Independent Set: Update of partial solutions



3 unions of $\leqslant d+2$ red connected subgraphs to consider in G_{i+1} with u, or v, or both

FO model checking on graphs of bounded twin-width

Generalization of the previous algorithm to:

Theorem (B., Kim, Thomassé, Watrigant '20)

FO model checking can be solved in time $f(|\varphi|, d) \cdot |V(G)|$ *on graphs* G *given with a* d*-sequence.*

Gaifman's locality + MSO model checking algorithm

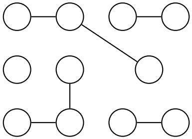
First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

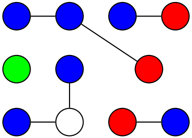
FO transduction: color by $O(1)$ unary relations, interpret, delete



First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

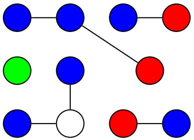
FO transduction: color by $O(1)$ unary relations, interpret, delete



First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete

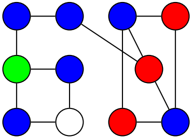


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee(R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete

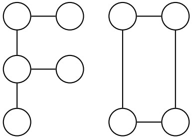


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee(R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

First-order interpretations and transductions

FO interpretation: redefine the edges by a first-order formula
$\varphi(x, y) = \neg E(x, y)$ (complement)
$\varphi(x, y) = E(x, y) \vee \exists z E(x, z) \wedge E(z, y)$ (square)

FO transduction: color by $O(1)$ unary relations, interpret, delete

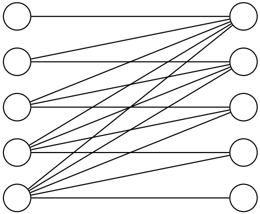


Stability and dependence of hereditary classes

Due to [Baldwin, Shelah '85; Braunfeld, Laskowski '22]

Stable class: no transduction of the class contains all ladders
Dependent class: no transduction of the class contains all graphs

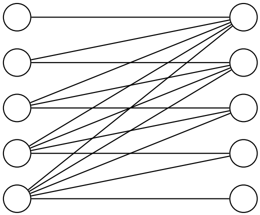
ladder



Stability and dependence of hereditary classes

Stable class: no transduction of the class contains all ladders
Dependent class: no transduction of the class contains all graphs

ladder

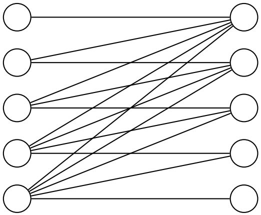


Bounded-degree graphs $\rightarrow$ stable
Unit interval graphs $\rightarrow$ dependent but not stable
Interval graphs $\rightarrow$ not dependent

Stability and dependence of hereditary classes

Stable class: no transduction of the class contains all ladders
Dependent class: no transduction of the class contains all graphs

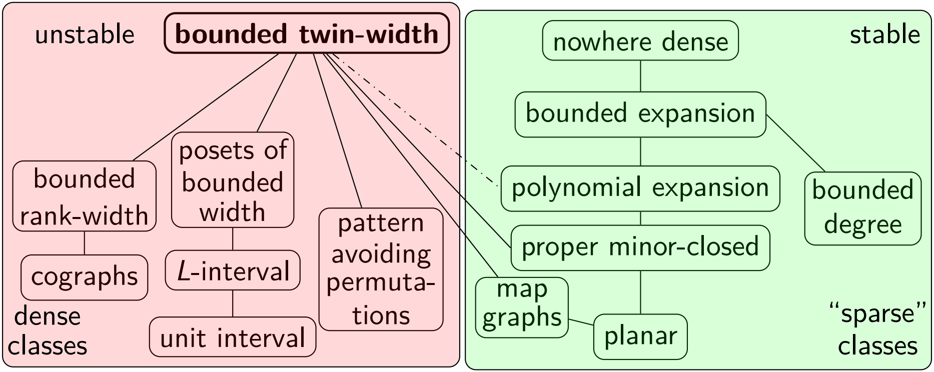
ladder



Bounded-degree graphs → stable
Unit interval graphs → dependent but not stable
Interval graphs → not dependent

Bounded twin-width classes → dependent, but in general not stable

Classes with known tractable FO model checking



FO MODEL CHECKING solvable in $f(|\varphi|, d)n$ on graphs with a d-sequence
[B., Kim, Thomassé, Watrigant '20]

First-order transductions preserve bounded twin-width

Theorem (B., Kim, Thomassé, Watrigant '20)

For every class $\mathcal{C}$ of binary structures with bounded twin-width and transduction $\mathscr{T}$, the class $\mathscr{T}(\mathcal{C})$ has bounded twin-width.

First-order transductions preserve bounded twin-width

Theorem (B., Kim, Thomassé, Watrigant '20)

For every class $\mathcal{C}$ of binary structures with bounded twin-width and transduction $\mathscr{T}$, the class $\mathscr{T}(\mathcal{C})$ has bounded twin-width.

- Making copies does not change the twin-width
- Adding a unary relation at most doubles it

First-order transductions preserve bounded twin-width

Theorem (B., Kim, Thomassé, Watrigant '20)

For every class $\mathcal{C}$ of binary structures with bounded twin-width and transduction $\mathscr{T}$, the class $\mathscr{T}(\mathcal{C})$ has bounded twin-width.

- Making copies does not change the twin-width
- Adding a unary relation at most doubles it
- Refine parts of the partition sequence by types

The lens of contraction sequences

Class of bounded	FO transduction of	constraint on red graphs	efficient MC
linear rank-width	linear order	bd #edges	MSO
rank-width	tree order	bd component	**MSO**
twin-width	**?**	bd degree	**FO**

Permutations strike back

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)
A class of binary structures has bounded twin-width if and only if it is a first-order transduction of a proper permutation class.

Theorem (B., Bourneuf, Geniet, Thomassé '24)
Pattern-free permutations are bounded products of separable permutations.

Permutations strike back

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)

A class of binary structures has bounded twin-width if and only if it is a first-order transduction of a proper permutation class.

Theorem (B., Bourneuf, Geniet, Thomassé '24)

There is a function f such that for every permutation σ, for every permutation τ of $Av(\sigma)$ there are t separable permutations $\sigma_1, \sigma_2 \ldots, \sigma_t$ with $t \leqslant f(|\sigma|)$ and $\tau = \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_t$.

Permutations strike back

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)

A class of binary structures has bounded twin-width if and only if it is a first-order transduction of a proper permutation class.

Theorem (B., Bourneuf, Geniet, Thomassé '24)

There is a function f such that for every permutation σ, for every permutation τ of $Av(\sigma)$ there are t separable permutations $\sigma_1, \sigma_2 \ldots, \sigma_t$ with $t \leqslant f(|\sigma|)$ and $\tau = \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_t$.

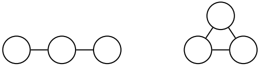
As a by-product of these two results,

Corollary (B., Bourneuf, Geniet, Thomassé '24)

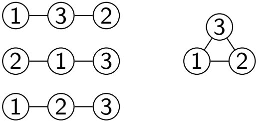
There is a proper permutation class $\mathcal{P}$ such that every class of binary structures has bounded twin-width if and only if it is a first-order transduction of $\mathcal{P}$.

Growth of Graph Classes

Number of unlabeled n-vertex graphs of $\mathcal{C}$ up to isomorphism, or
Number of labeled n-vertex graphs of $\mathcal{C}$

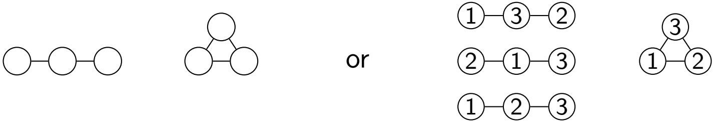


or



Growth of Graph Classes

Number of unlabeled n-vertex graphs of $\mathcal{C}$ up to isomorphism, or
Number of labeled n-vertex graphs of $\mathcal{C}$



Small: labeled growth $n!2^{O(n)}$

Tiny: unlabeled growth $2^{O(n)}$

Small and tiny classes

Theorem (B., Geniet, Kim, Thomassé, Watrigant '21)
Classes of bounded twin-width are small.

And even,

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)
Classes of bounded twin-width are tiny.

Unifies the Marcus–Tardos(–Klazar) theorem and the same statement for classes excluding a minor

Small and tiny classes

Theorem (B., Geniet, Kim, Thomassé, Watrigant '21)

Classes of bounded twin-width are small.

And even,

Theorem (B., Nešetřil, Ossona de Mendez, Siebertz, Thomassé '21)

Classes of bounded twin-width are tiny.

Unifies the Marcus–Tardos(–Klazar) theorem and the same statement for classes excluding a minor

Could the converse hold for hereditary classes?

Twin-width of groups

For a finitely-generated group:
sup of the twin-width of the age of its Cayley graph

Twin-width of a group action
$\phi : \Gamma \to \mathrm{Bij}(X)$ and $g \in \Gamma$:
k_g, minimum grid number of the permutation matrix $M^{<}_{\phi(g)}$

Finite twin-width: for every $g \in \Gamma$, k_g is finite
Finite uniform twin-width: $\exists t$ s.t. for every $g \in \Gamma$, $k_g \leqslant t$

Twin-width of a group: use action of Γ on itself by left product

Finite and infinite twin-width

Examples of groups with finite twin-width:
Abelian, hyperbolic, orderable, solvable, polynomial growth, etc.

Finite and infinite twin-width

Examples of groups with finite twin-width:
Abelian, hyperbolic, orderable, solvable, polynomial growth, etc.

Theorem (B., Geniet, Tessera, Thomassé '22)

There is a finitely-generated group with infinite twin-width.

Small hereditary class of unbounded twin-width

Ordered binary structures

Theorem (B., Giocanti, Ossona de Mendez, Simon, Thomassé, Toruńczyk '22)

Let $\mathscr{C}$ be a hereditary class of ordered graphs. The following are equivalent.

(1) *$\mathscr{C}$ has bounded twin-width.*

(2) *$\mathscr{C}$ is dependent.*

(3) *$\mathscr{C}$ contains $2^{O(n)}$ ordered n-vertex graphs.*

(4) *$\mathscr{C}$ contains less than $\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} k!$ ordered n-vertex graphs, for some n.*

(5) *$\mathscr{C}$ does not include one of 25 hereditary ordered graph classes with unbounded twin-width.*

(6) *FO-model checking is fixed-parameter tractable on $\mathscr{C}$.*

Open questions

- Algorithm to compute/approximate twin-width
- Constructions of bounded-degree graphs of unbounded twin-width
- Common generalization with stable classes (see flip-width of Szymon Toruńczyk)
- Dividing line bounded/unbounded twin-width in groups
- Separation of finite twin-width and finite uniform twin-width
- Generalization to higher-arity relations
- Is small and tiny equivalent for hereditary classes?

Open questions

- Algorithm to compute/approximate twin-width
- Constructions of bounded-degree graphs of unbounded twin-width
- Common generalization with stable classes (see flip-width of Szymon Toruńczyk)
- Dividing line bounded/unbounded twin-width in groups
- Separation of finite twin-width and finite uniform twin-width
- Generalization to higher-arity relations
- Is small and tiny equivalent for hereditary classes?

Thank you for your attention!