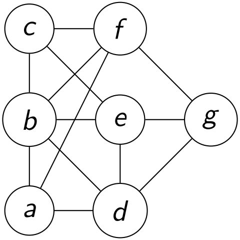
Twin-width and Logic

Édouard Bonnet

ENS Lyon, LIP

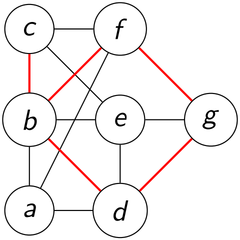
November 6th, combprob2023, Leeds, UK

Graphs



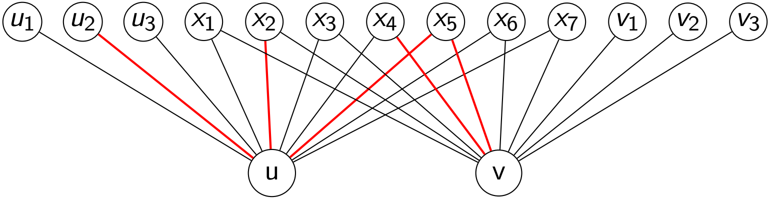
Two outcomes between a pair of vertices:
edge or non-edge

Trigraphs



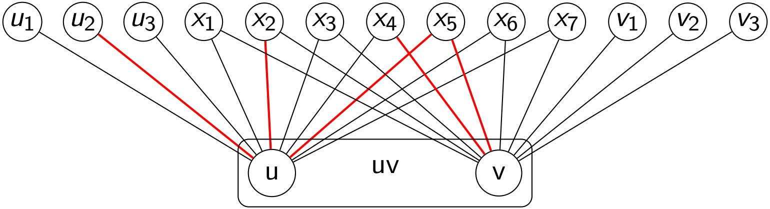
Three outcomes between a pair of vertices:
edge, or non-edge, or red edge (error edge)

Contractions in trigraphs



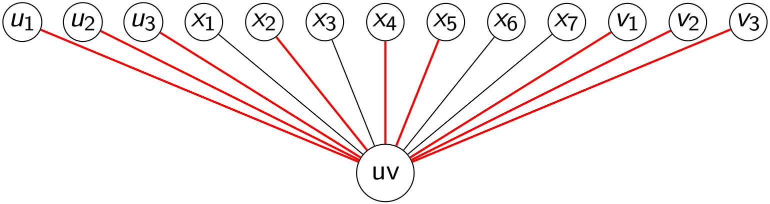
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



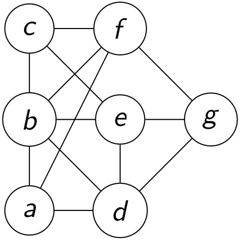
Identification of two non-necessarily adjacent vertices

Contractions in trigraphs



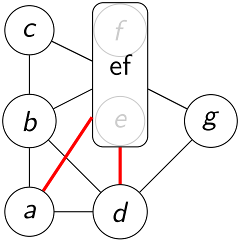
edges to $N(u) \triangle N(v)$ turn red, for $N(u) \cap N(v)$ red is absorbing

Contraction sequence



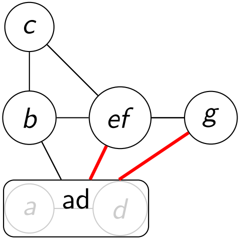
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



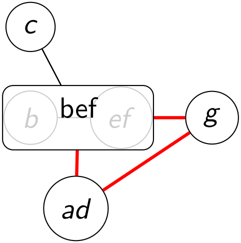
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



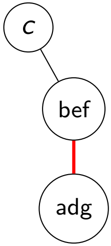
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

Contraction sequence



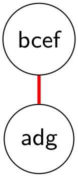
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

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



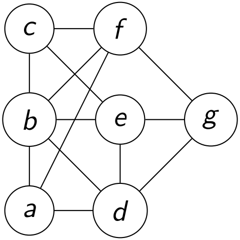
A contraction sequence of G:
Sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_2, G_1$ such that
G_i is obtained by performing one contraction in G_{i+1}.

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}.

Twin-width

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

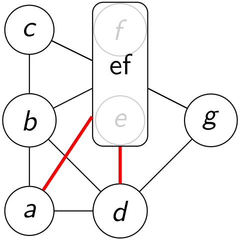


Maximum red degree $= 0$

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.

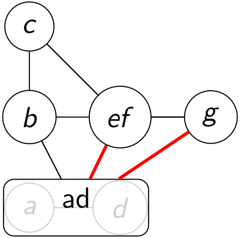


Maximum red degree = 2

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.

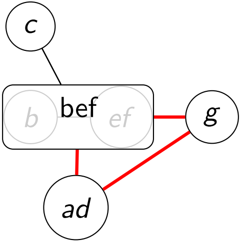


Maximum red degree $= 2$

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.

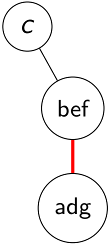


Maximum red degree = 2

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.

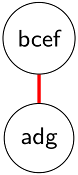


Maximum red degree $= 1$

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

- 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.

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.*

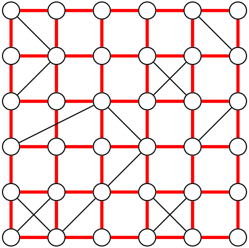
Theorem (B., Geniet, Kim, Thomassé, Watrigant '20 & '21)

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

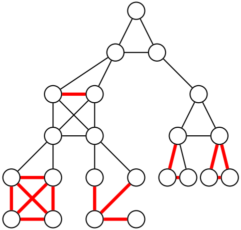
- *Bounded rank-width, and even, boolean-width graphs,*
- *every hereditary proper subclass of permutation graphs,*
- *posets of bounded antichain size (seen as digraphs),*
- *unit interval graphs,*
- *K_t-minor free graphs,*
- *map graphs,*
- *subgraphs of d-dimensional grids,*
- *K_t-free unit d-dimensional 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.*

Can we solve problems faster, given an $O(1)$-sequence?

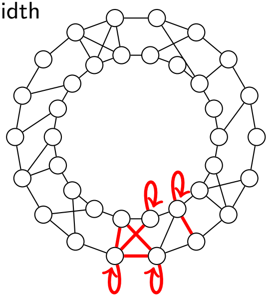
Different conditions imposed in the sequence of red graphs



bd degree: defines bd twin-width



bd component: redefines bd cliquewidth



bd #edges: redefines bd linear cliquewidth

Formulas, sentences, and 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: $G \models \varphi$?

Formulas, sentences, and 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:

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

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

Formulas, sentences, and 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

Formulas, sentences, and 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:

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

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

Formulas, sentences, and 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

Formulas, sentences, and 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:

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

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

Formulas, sentences, and 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:

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

$G \models \varphi$? $\Leftrightarrow$ 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*.*

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.

- as the incidence graph preserves bounded treewidth, possible edge-set quantification
- linear FPT approximation for treewidth
- (polynomial) FPT approximation for clique-width

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:

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

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

Theorem (folklore)

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

Proof.

"$\mathcal{L}[k+1]$ are Boolean combinations of $\exists x \mathcal{L}[k]$." □

Rank-k m-types

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

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

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

Theorem (folklore)

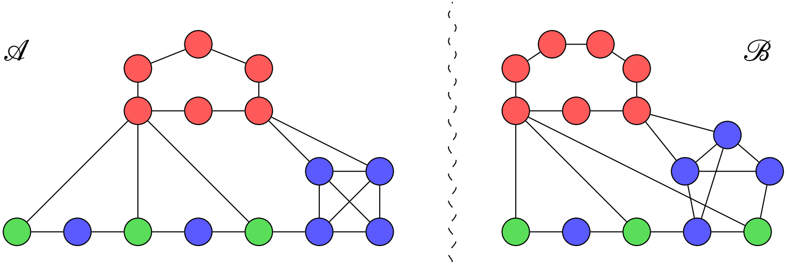
For $\mathcal{L} \in \{FO, MSO\}$, the number of rank-$k$ m-types is bounded by a function of k and m only.

Proof.

"$\mathcal{L}[k+1]$ are Boolean combinations of $\exists x \mathcal{L}[k]$." □

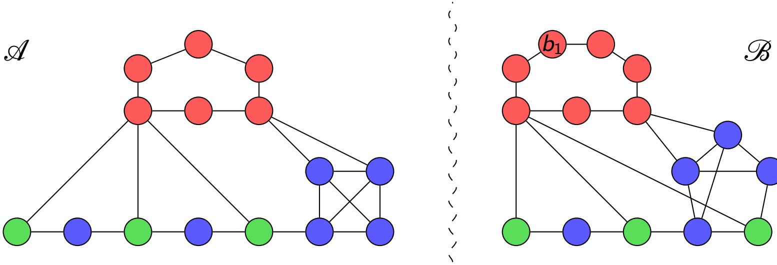
Rank-k types partition the graphs into $g(k)$ classes.
Efficient Model Checking = quickly finding the class of the input.

FO Ehrenfeucht-Fraissé game



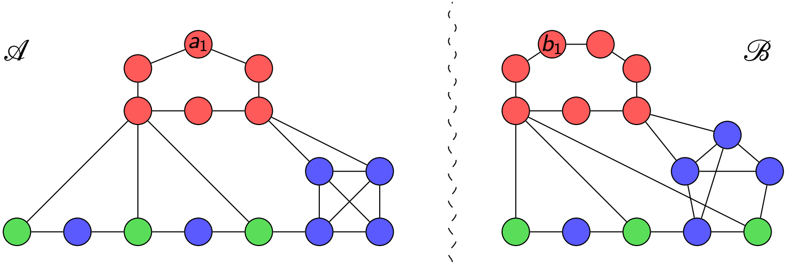
2-player game on two σ-structures $\mathscr{A}, \mathscr{B}$ (for us, colored graphs)

FO Ehrenfeucht-Fraissé game



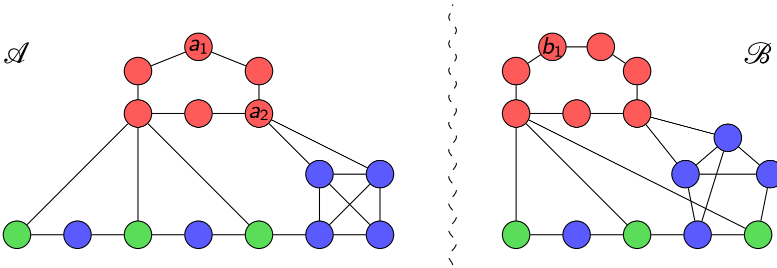
At each round, Spoiler picks a structure ($\mathscr{B}$) and a vertex therein

FO Ehrenfeucht-Fraissé game



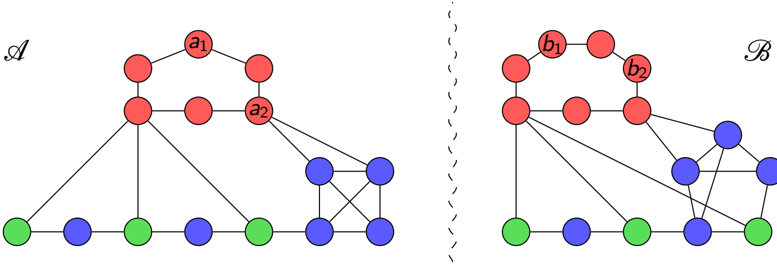
Duplicator answers with a vertex in the other structure

FO Ehrenfeucht-Fraissé game



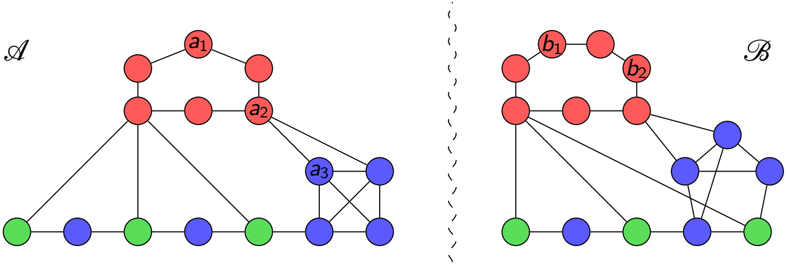
After q rounds, Duplicator wishes that $a_i \mapsto b_i$ is an isomorphism between $\mathscr{A}[a_1, \ldots, a_k]$ and $\mathscr{B}[b_1, \ldots, b_k]$

FO Ehrenfeucht-Fraissé game



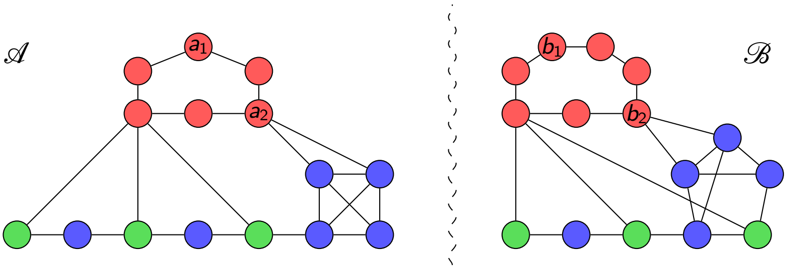
After q rounds, Duplicator wishes that $a_i \mapsto b_i$ is an isomorphism between $\mathscr{A}[a_1, \ldots, a_k]$ and $\mathscr{B}[b_1, \ldots, b_k]$

FO Ehrenfeucht-Fraissé game



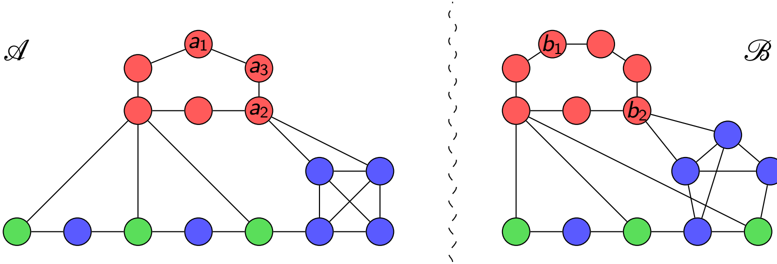
When no longer possible, Spoiler wins

FO Ehrenfeucht-Fraissé game



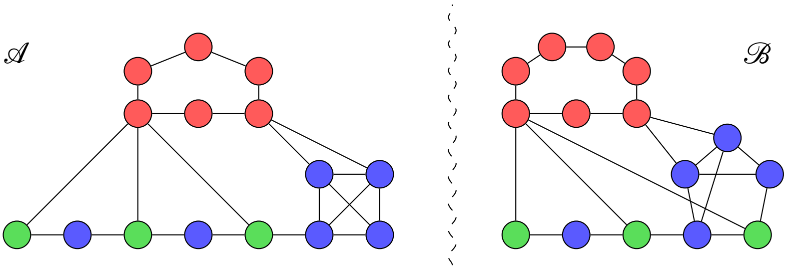
When no longer possible, Spoiler wins

FO Ehrenfeucht-Fraissé game



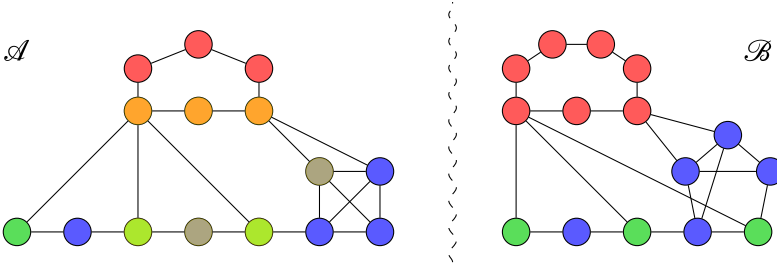
If Duplicator can survive k rounds, we write $\mathscr{A} \equiv_k^{\mathsf{FO}} \mathscr{B}$
Here $\mathscr{A} \equiv_2^{\mathsf{FO}} \mathscr{B}$ and $\mathscr{A} \not\equiv_3^{\mathsf{FO}} \mathscr{B}$

MSO Ehrenfeucht-Fraissé game



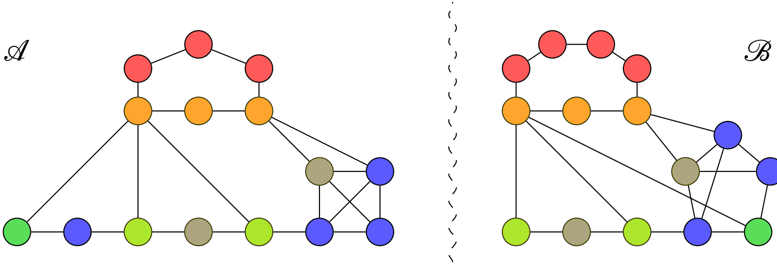
Same game but Spoiler can now play set moves

MSO Ehrenfeucht-Fraissé game



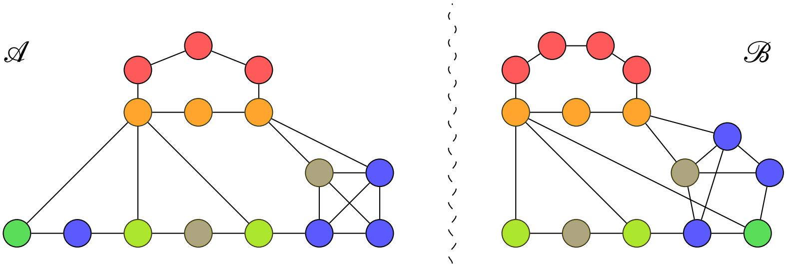
Same game but Spoiler can now play set moves

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-Fraissé)

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

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

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 $(E, U_1, \dots, U_d)$-structures, interpreted as a graph vertex partitioned in d parts

Maintain for every red component C of every trigraph G_i

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

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

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

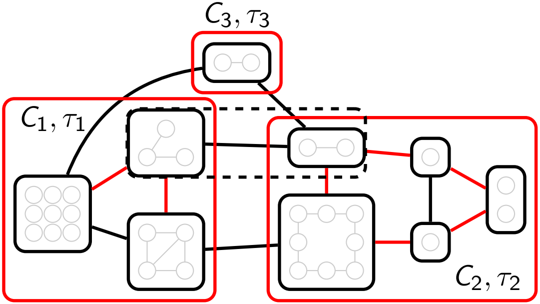
For each $v \in V(G)$, $\mathrm{tp}_k(G, \mathcal{P}_n, \{v\}) =$ type of K_1
$\mathrm{tp}_k(G, \mathcal{P}_1, \{V(G)\}) =$ type of G

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

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



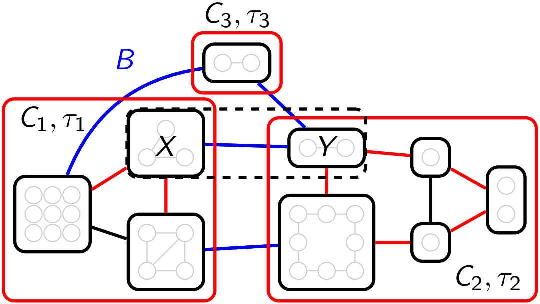
$\tau = \mathrm{tp}_k^{\mathrm{MSO}}(G, \mathcal{P}_i, C)$ based on the $\tau_j = \mathrm{tp}_k^{\mathrm{MSO}}(G, \mathcal{P}_{i+1}, C_j)$?

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

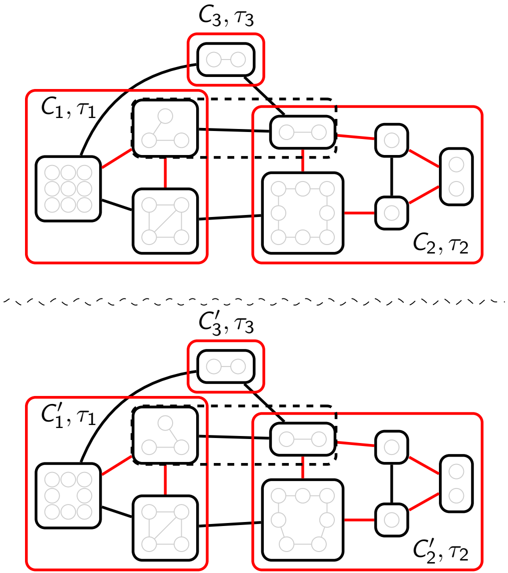
Maintain for every red component C of every trigraph G_i

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



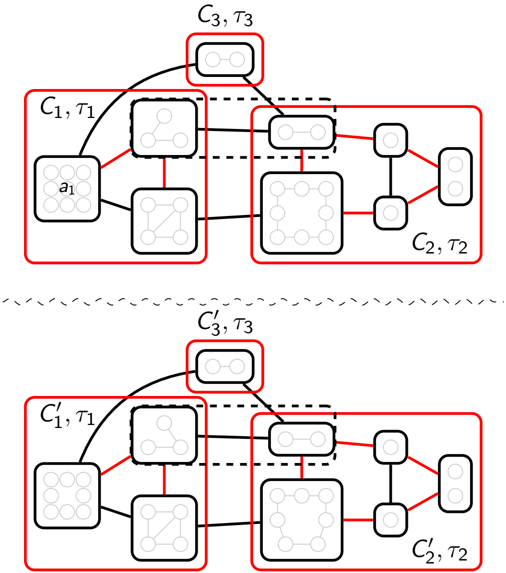
C arises from $C_1, \ldots, C_{d'}$: $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$

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



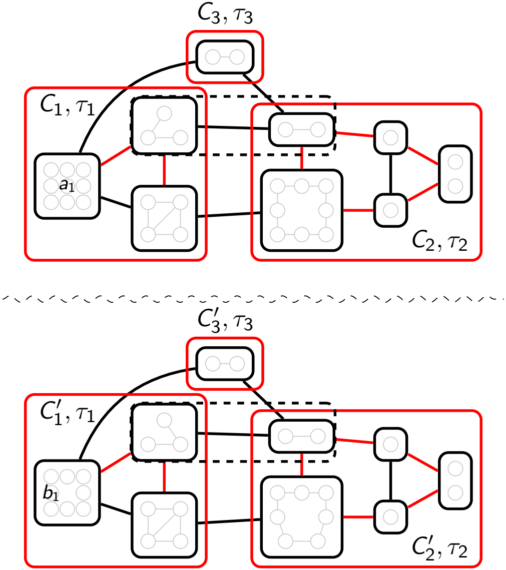
Duplicator combines her strategies in the red components

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



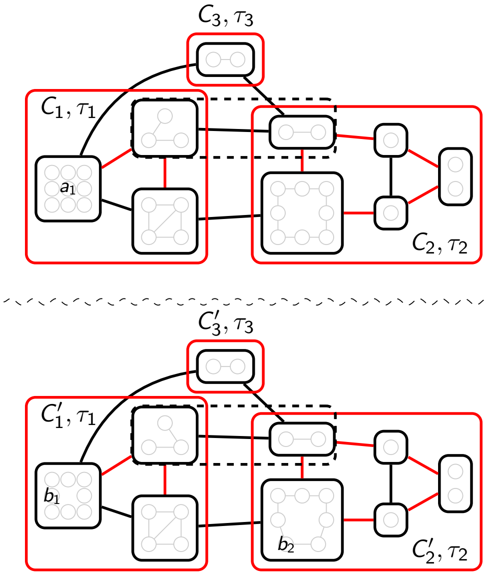
If Spoiler plays a vertex in the component of type τ_1,

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



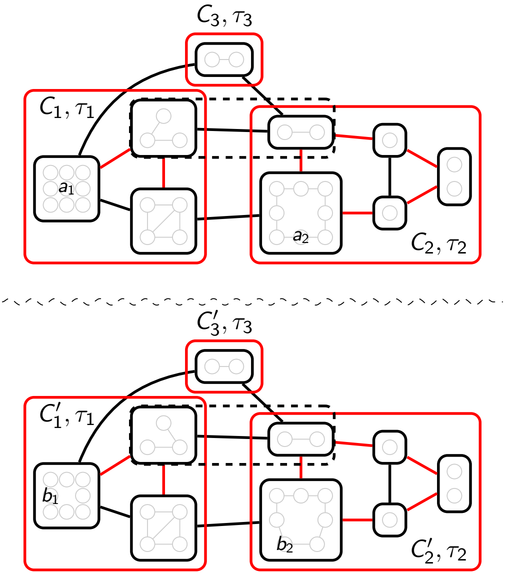
Duplicator answers the corresponding winning move

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



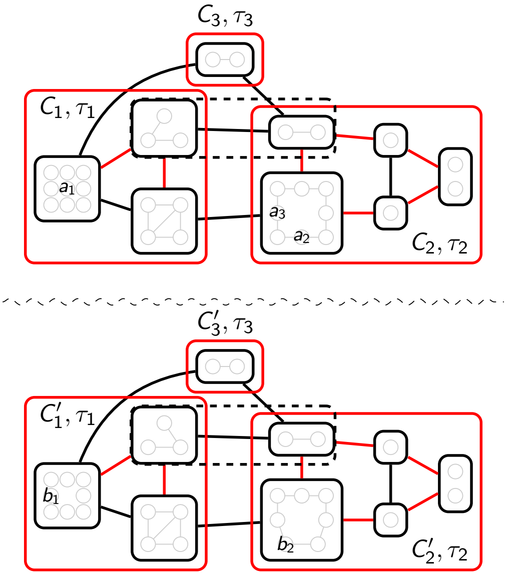
Same in the component of type τ_2

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



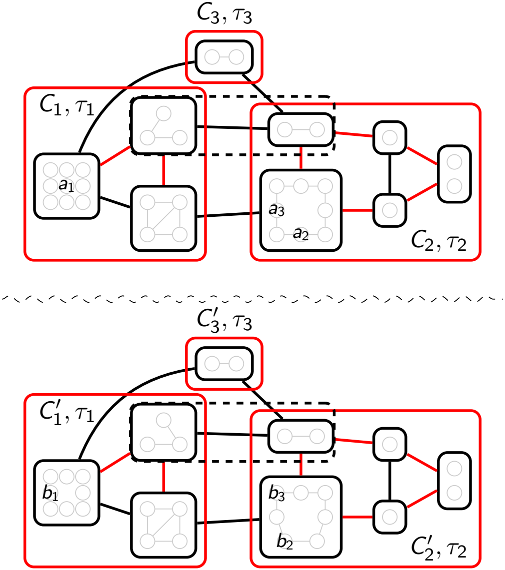
Same in the component of type τ_2

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



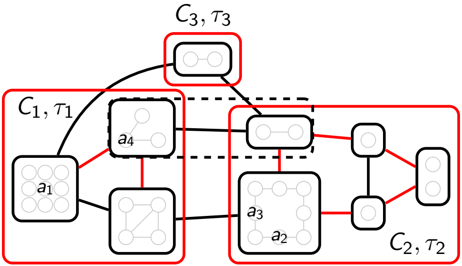
Same in the component of type τ_2

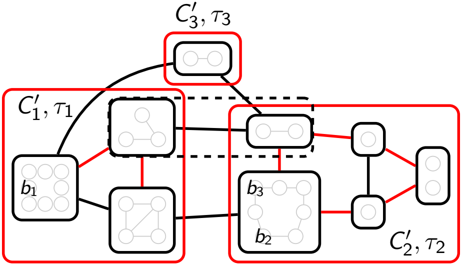
Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



Same in the component of type τ_2

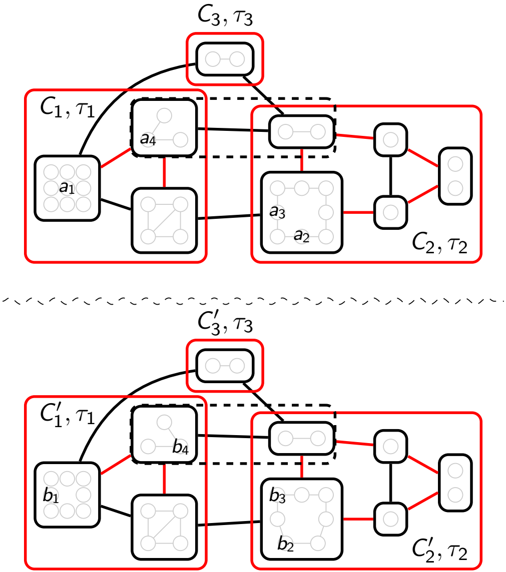
Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game





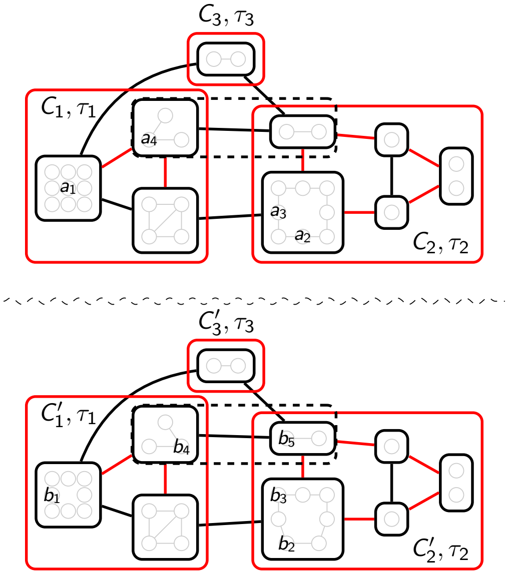
and so on

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



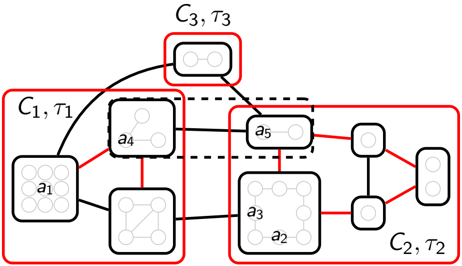
and so on

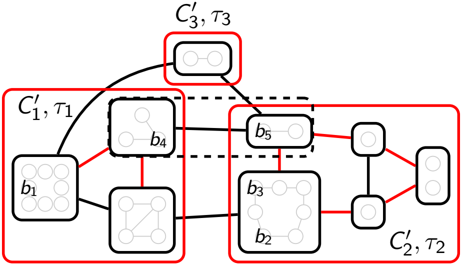
Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game



and so on

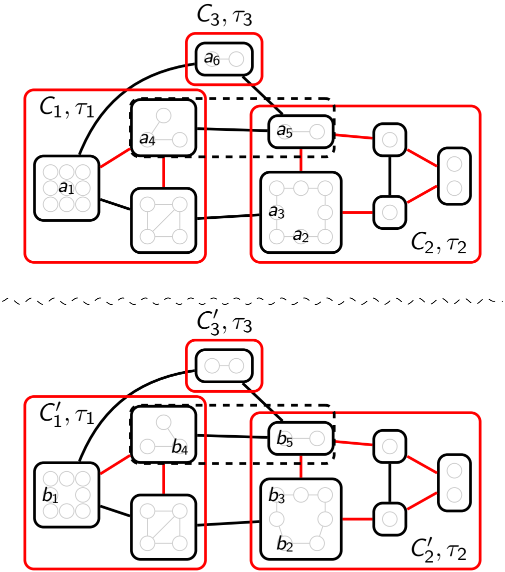
Showing $\tau = F(\tau_1, \ldots, \tau_{d'}, B, X, Y)$ via MSO EF game





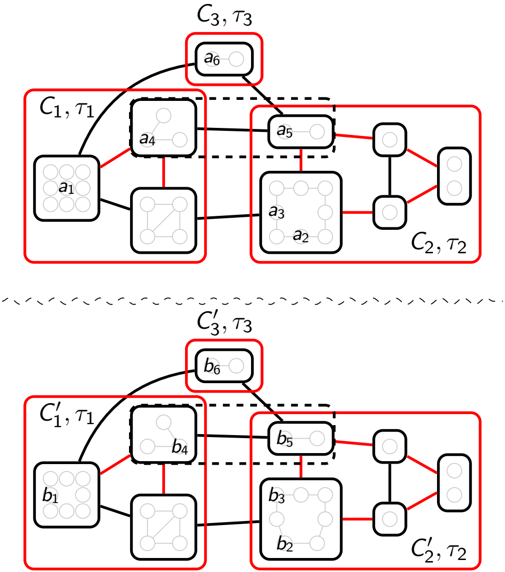
and so on

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



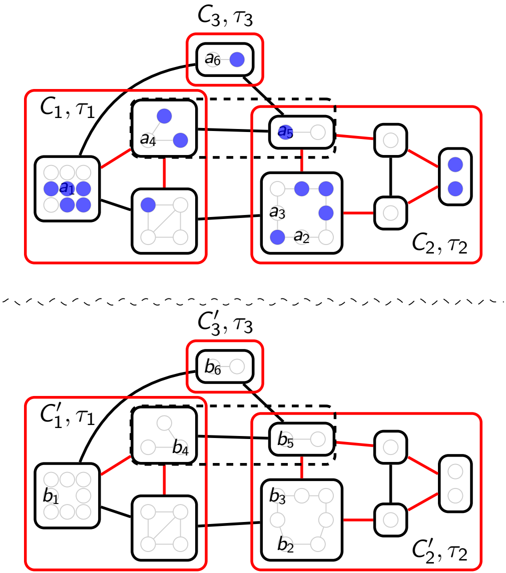
and so on

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



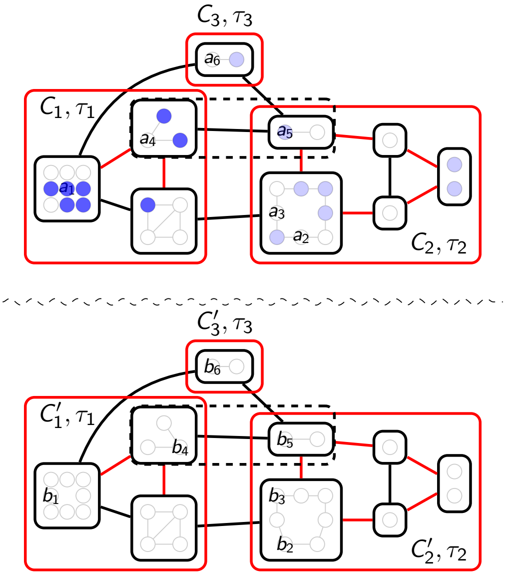
and so on

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



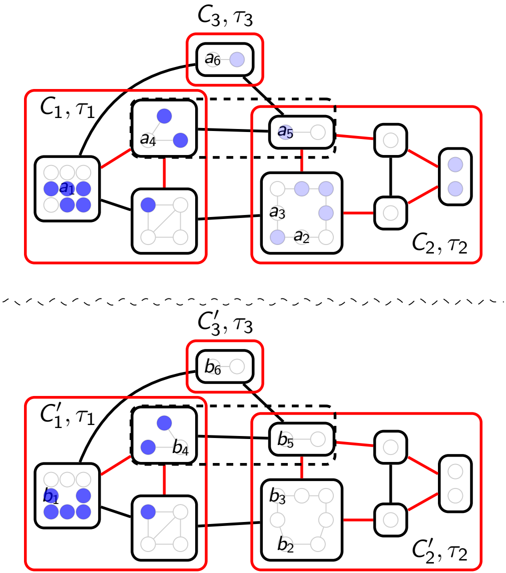
If Spoiler plays a set, Duplicator looks at the intersection with C_1,

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



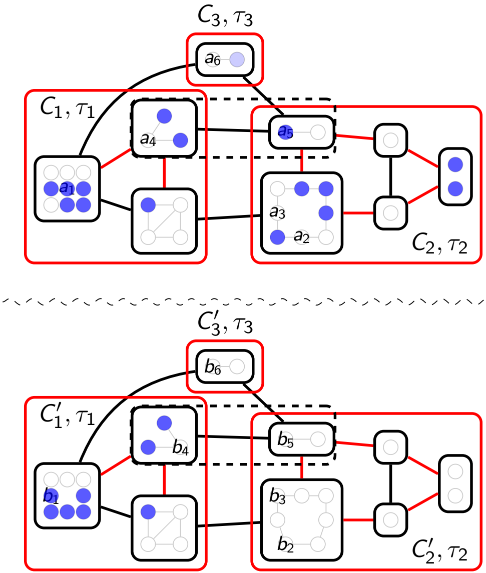
If Spoiler plays a set, Duplicator looks at the intersection with C_1,

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



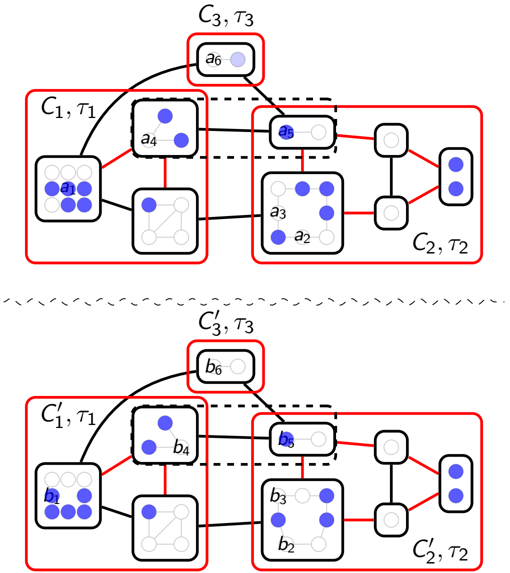
calls her winning strategy in C'_1

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



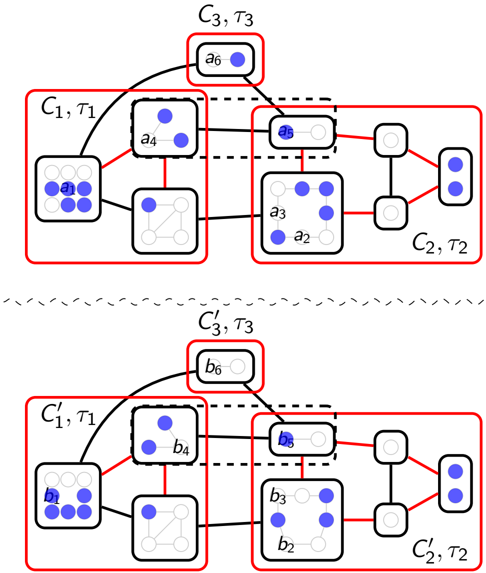
same for the other components

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



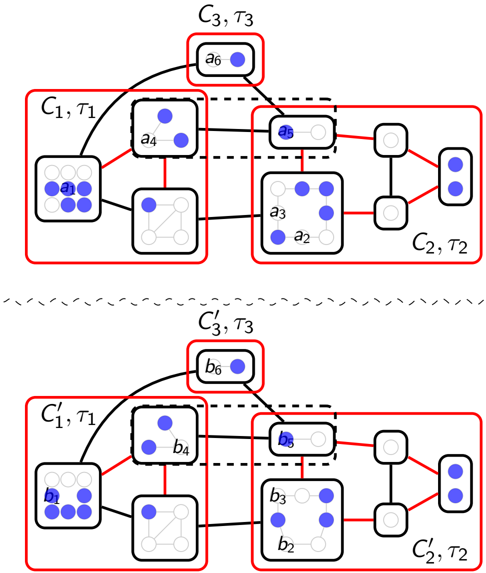
same for the other components

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



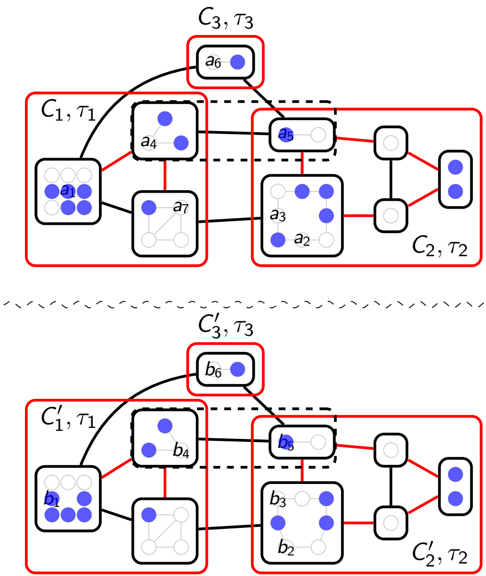
same for the other components

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



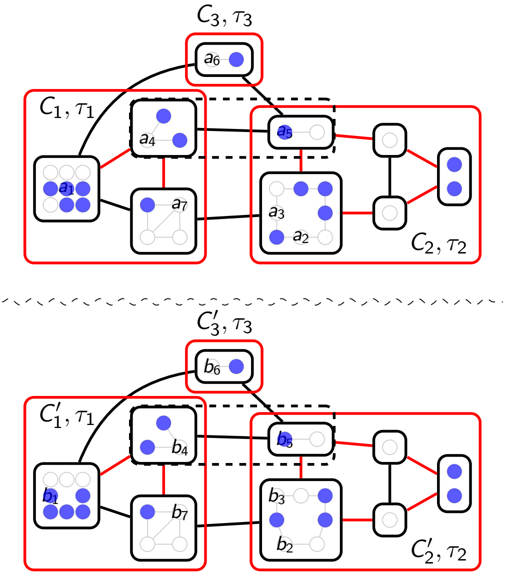
and plays the union

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



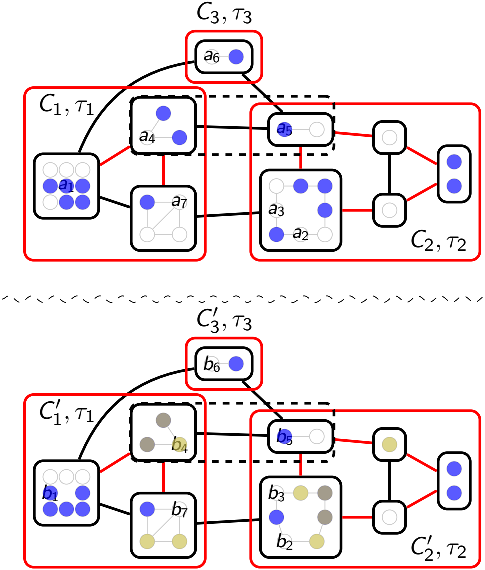
that fully defines the winning strategy of Duplicator

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



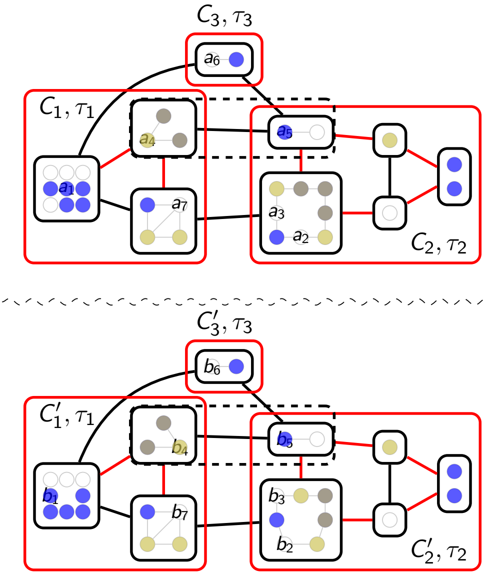
that fully defines the winning strategy of Duplicator

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



that fully defines the winning strategy of Duplicator

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

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, 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

d-sequence: $G = G_n, G_{n-1}, \dots, G_2, G_1 = K_1$

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, store in $T[D, i]$ one largest independent set in $G\langle D \rangle$ intersecting every vertex of D.**

Initialization: $T[\{v\}, n] = \{v\}$

End: $T[\{V(G)\}, 1] =$ IS of size at least k or largest IS in G

Running time: $d^{2k}n^2$ red connected subgraphs,
actually only $d^{2k}n = 2^{O_d(k)}n$ updates

k-Independent Set given a d-sequence

d-sequence: $G = G_n, G_{n-1}, \ldots, G_2, G_1 = K_1$

Algorithm: **For every connected subset D of size at most k of the red graph of every G_i, store in $T[D, i]$ one largest independent set in $G\langle D \rangle$ intersecting every vertex of D.**

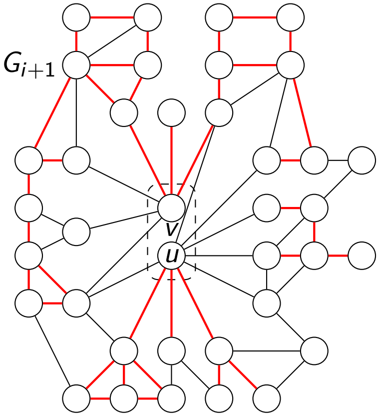
Initialization: $T[\{v\}, n] = \{v\}$

End: $T[\{V(G)\}, 1] =$ IS of size at least k or largest IS in G

Running time: $d^{2k}n^2$ red connected subgraphs,
actually only $d^{2k}n = 2^{O_d(k)}n$ updates

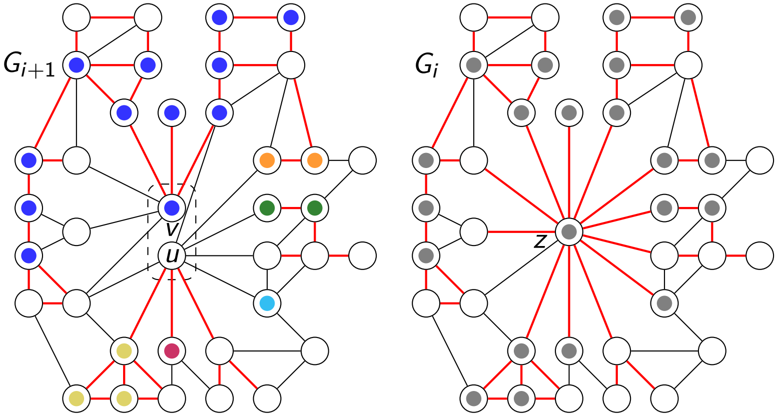
How to compute $T[D, i]$ from all the $T[D', i+1]$?

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

We will now generalize 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.

FO model checking on graphs of bounded twin-width

We will now generalize the previous algorithm to:

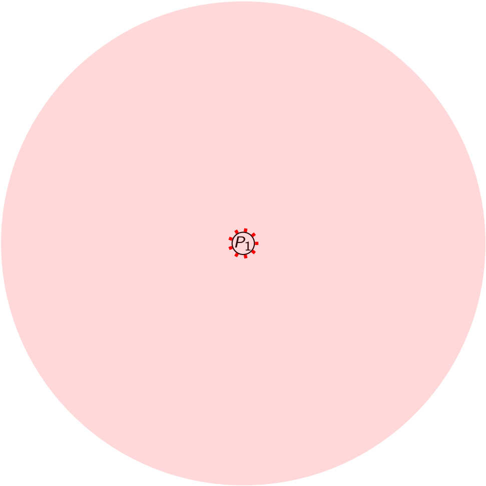
Theorem (B., Kim, Thomassé, Watrigant '20)

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

Add **Gaifman's locality** to our MSO model checking algorithm

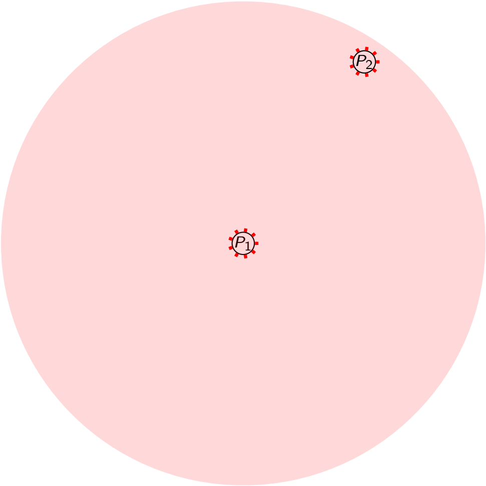
Following [Gajarský, Pilipczuk, Przybyszewski, Toruńczyk '22]

Local tuple of parts



$(P_1, P_2, \ldots, P_q)$ is *k-local around P_1* in $(G, \mathcal{P}_i)$ if...

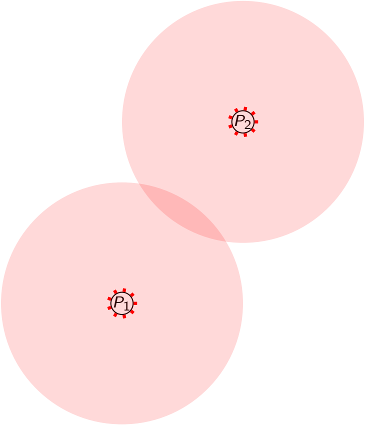
Local tuple of parts



$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...

P_2 is at distance at most 2^{k-2} from $\{P_1\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

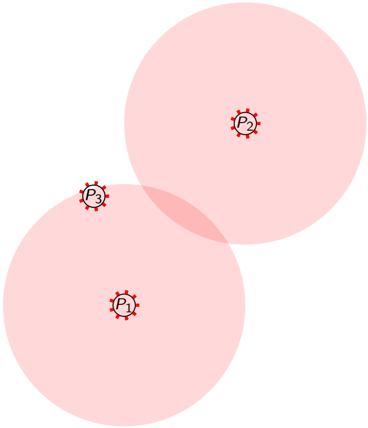
Local tuple of parts



$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...

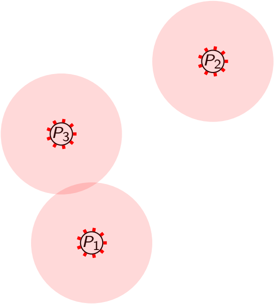
P_2 is at distance at most 2^{k-2} from $\{P_1\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



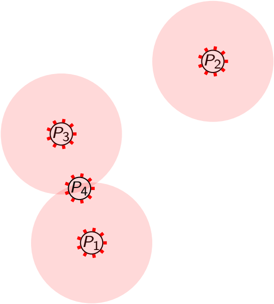
$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_3 is at distance at most 2^{k-3} from $\{P_1, P_2\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



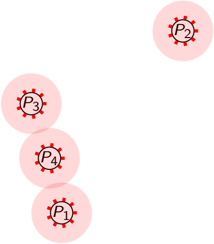
$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_3 is at distance at most 2^{k-3} from $\{P_1, P_2\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



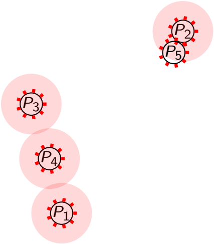
$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_4 is at distance at most 2^{k-4} from $\{P_1, P_2, P_3\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



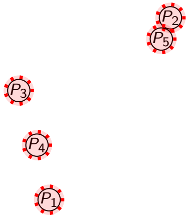
$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_4 is at distance at most 2^{k-4} from $\{P_1, P_2, P_3\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



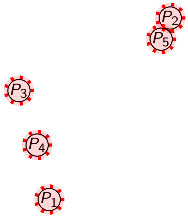
$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_q is at distance at most 2^{k-q} from $\{P_1, ..., P_{q-1}\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_q is at distance at most 2^{k-q} from $\{P_1, ..., P_{q-1}\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Local tuple of parts



$(P_1, P_2, \ldots, P_q)$ is *k-local around* P_1 in $(G, \mathcal{P}_i)$ if...
P_q is at distance at most 2^{k-q} from $\{P_1, \ldots, P_{q-1}\}$ in $\mathcal{R}(G, \mathcal{P}_i)$

Partitioned local sentences and types

A prenex sentence is *partitioned local around* X in $(G, \mathcal{P}_i)$ if of the form $Qx_1 \in X \; Qx_2 \in P_2 \; \ldots \; Qx_k \in P_k \; \psi(x_1, \ldots, x_k)$ with

- ψ is quantifier-free, and
- $(X, P_2, \ldots, P_k)$ local around X in $(G, \mathcal{P}_i)$.

Partitioned local sentences and types

A prenex sentence is *partitioned local around* X in $(G, \mathcal{P}_i)$ if of the form $Qx_1 \in X \ Qx_2 \in P_2 \ \dots \ Qx_k \in P_k \ \psi(x_1, \dots, x_k)$ with

- ψ is quantifier-free, and
- $(X, P_2, \dots, P_k)$ local around X in $(G, \mathcal{P}_i)$.

And the corresponding types:

$$\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \{\varphi : \mathsf{qr}(\varphi) \leqslant k,$$

$$\varphi \text{ is partitioned local around } X \text{ in } (G, \mathcal{P}_i),$$

$$(G, \mathcal{P}_i) \models \varphi\}.$$

Partitioned local sentences/types in $(G, \mathcal{P}_n)$ and $(G, \mathcal{P}_1)$

Initialization of the dynamic programming

In $(G, \mathcal{P}_n = \{\{v\} : v \in V(G)\})$: for every $v \in V(G)$,
$Qx_1 \in \{v\}\ Qx_2 \in \{v\}\ \ldots\ Qx_k \in \{v\}\ \psi \equiv \psi(v, v, \ldots, v)$

Partitioned local types are easy to compute in $(G, \mathcal{P}_n)$

Partitioned local sentences/types in $(G, \mathcal{P}_n)$ and $(G, \mathcal{P}_1)$

Initialization of the dynamic programming

In $(G, \mathcal{P}_n = \{\{v\} : v \in V(G)\})$: for every $v \in V(G)$,
$Qx_1 \in \{v\}\ Qx_2 \in \{v\}\ \ldots\ Qx_k \in \{v\}\ \psi \equiv \psi(v, v, \ldots, v)$

Partitioned local types are easy to compute in $(G, \mathcal{P}_n)$

Output of the dynamic programming

In $(G, \mathcal{P}_1 = \{V(G)\})$:
$Qx_1 \in V(G)\ Qx_2 \in V(G)\ \ldots\ Qx_k \in V(G)\ \psi \equiv$ classic sentences

The partitioned local type in $(G, \mathcal{P}_1)$ coincides with the type of G

Partitioned local types give the partitioned types

Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \mathsf{ltp}_k^{\mathsf{FO}}(G', \mathcal{P}'_i, f(X))$

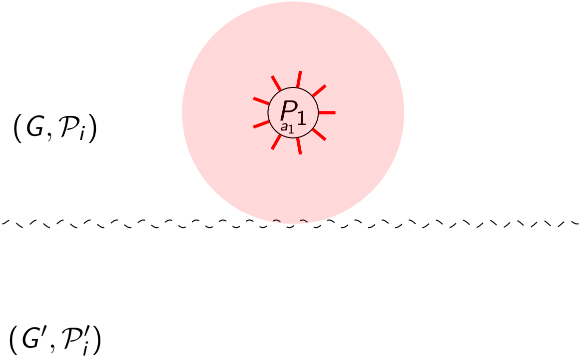
$(G, \mathcal{P}_i)$

$(G', \mathcal{P}'_i)$

Local strategies win the global game

Partitioned local types give the partitioned types

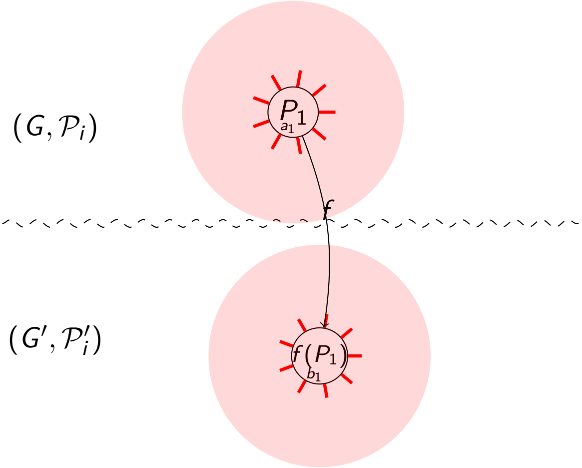
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \mathsf{ltp}_k^{\mathsf{FO}}(G', \mathcal{P}'_i, f(X))$



Say, Spoiler plays in P_1

Partitioned local types give the partitioned types

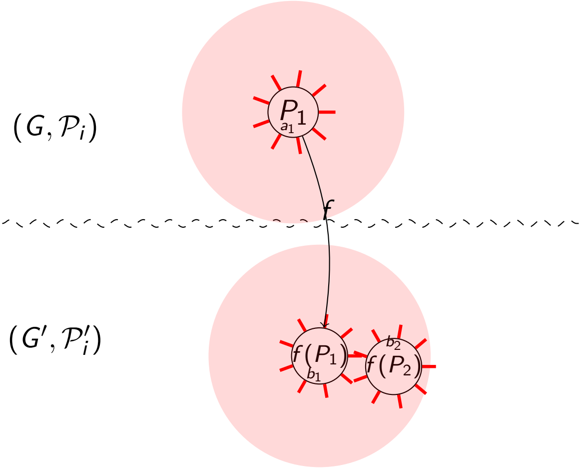
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}^{\mathsf{FO}}_k(G, \mathcal{P}_i, X) = \mathsf{ltp}^{\mathsf{FO}}_k(G', \mathcal{P}'_i, f(X))$



Duplicator answers in $f(P_1)$ following the local game around P_1

Partitioned local types give the partitioned types

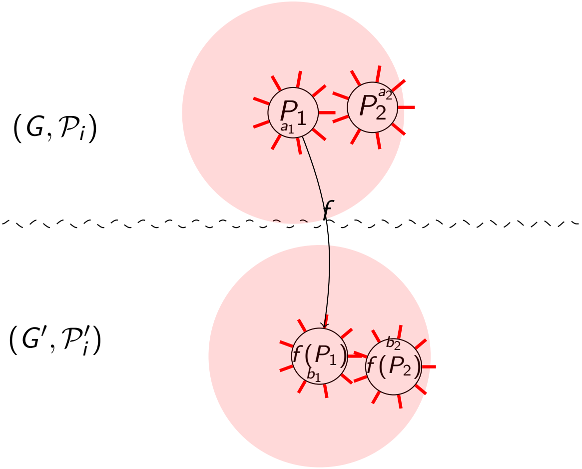
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathrm{ltp}_k^{\mathrm{FO}}(G, \mathcal{P}_i, X) = \mathrm{ltp}_k^{\mathrm{FO}}(G', \mathcal{P}'_i, f(X))$



Now when Spoiler plays close to P_1 or $f(P_1)$

Partitioned local types give the partitioned types

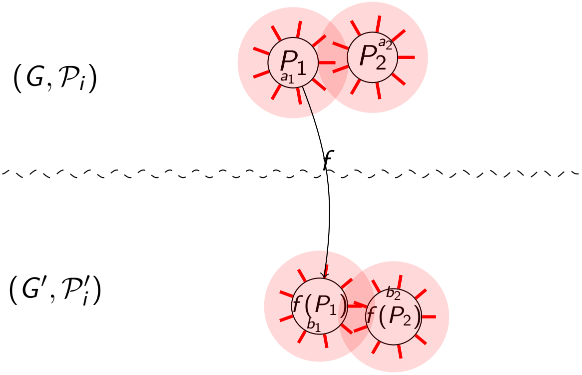
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \mathsf{ltp}_k^{\mathsf{FO}}(G', \mathcal{P}'_i, f(X))$



Duplicator follows the winning local strategy

Partitioned local types give the partitioned types

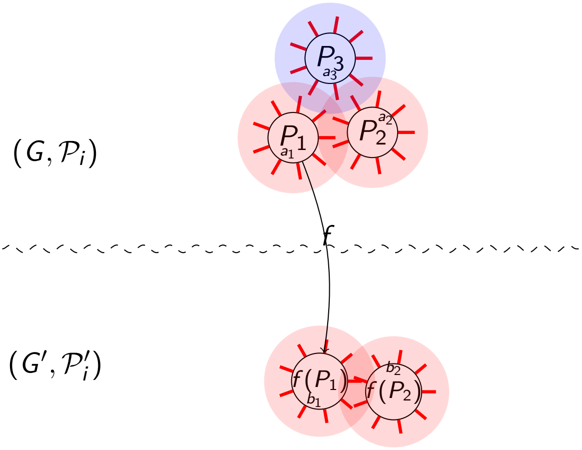
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \mathsf{ltp}_k^{\mathsf{FO}}(G', \mathcal{P}'_i, f(X))$



Duplicator follows the winning local strategy

Partitioned local types give the partitioned types

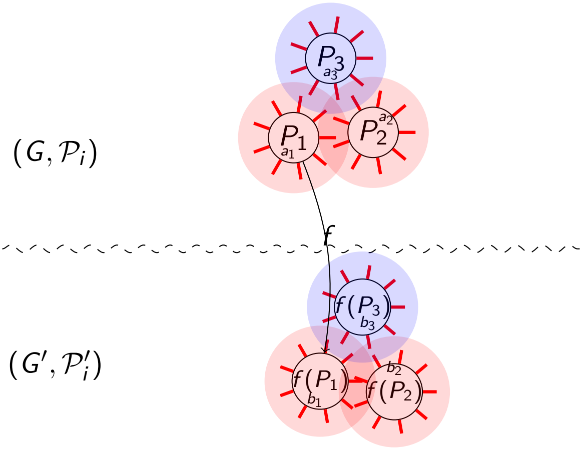
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}^{\mathsf{FO}}_k(G, \mathcal{P}_i, X) = \mathsf{ltp}^{\mathsf{FO}}_k(G', \mathcal{P}'_i, f(X))$



If Spoiler plays too far

Partitioned local types give the partitioned types

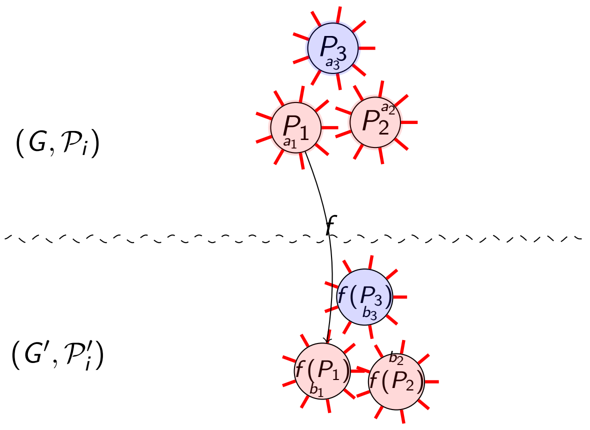
Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathsf{ltp}_k^{\mathsf{FO}}(G, \mathcal{P}_i, X) = \mathsf{ltp}_k^{\mathsf{FO}}(G', \mathcal{P}'_i, f(X))$



Duplicator starts a new local game around that new part

Partitioned local types give the partitioned types

Isom. $f : \mathcal{P}_i \to \mathcal{P}'_i$ with $\mathrm{ltp}_k^{\mathrm{FO}}(G, \mathcal{P}_i, X) = \mathrm{ltp}_k^{\mathrm{FO}}(G', \mathcal{P}'_i, f(X))$



Duplicator starts a new local game around that new part

Concluding as in the MSO model checking algorithm

$(G, \mathcal{P}_{i+1}) \rightsquigarrow (G, \mathcal{P}_i)$: X and Y are merged in Z

Partitioned local types around P

- only needs an update if P is at distance at most 2^{k-1} from Z

Concluding as in the MSO model checking algorithm

$(G, \mathcal{P}_{i+1}) \rightsquigarrow (G, \mathcal{P}_i)$: X and Y are merged in Z

Partitioned local types around P

- only needs an update if P is at distance at most 2^{k-1} from Z
- update only involves parts at distance at most 2^{k-1} from P

Concluding as in the MSO model checking algorithm

$(G, \mathcal{P}_{i+1}) \rightsquigarrow (G, \mathcal{P}_i)$: X and Y are merged in Z

Partitioned local types around P

- only needs an update if P is at distance at most 2^{k-1} from Z
- update only involves parts at distance at most 2^{k-1} from P
- hence at most d^{2^k} parts: conclude like MSO model checking

Concluding as in the MSO model checking algorithm

$(G, \mathcal{P}_{i+1}) \rightsquigarrow (G, \mathcal{P}_i)$: X and Y are merged in Z

Partitioned local types around P

- only needs an update if P is at distance at most 2^{k-1} from Z
- update only involves parts at distance at most 2^{k-1} from P
- hence at most d^{2^k} parts: conclude like MSO model checking

Each contraction: $O_{d,k}(1) = O(d^{2^k})$ updates in $O_{d,k}(1) = f(d, k)$
Total time: $O_{d,k}(n)$

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)

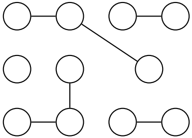
First-order interpretations and transductions

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)

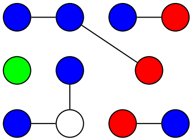
FO transduction: color by $O(1)$ unary relations, interpret, delete



First-order interpretations and transductions

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

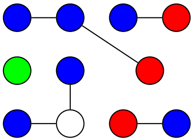
FO transduction: color by $O(1)$ unary relations, interpret, delete



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

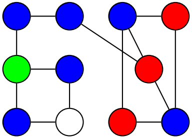


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg\exists z R(z) \wedge E(y, z))$$
$$\vee (R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg\exists z B(z) \wedge E(y, z))$$

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

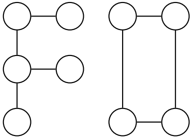


$$\varphi(x, y) = E(x, y) \vee (G(x) \wedge B(y) \wedge \neg \exists z R(z) \wedge E(y, z))$$
$$\vee (R(x) \wedge B(y) \wedge \exists z R(z) \wedge E(y, z) \wedge \neg \exists z B(z) \wedge E(y, z))$$

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

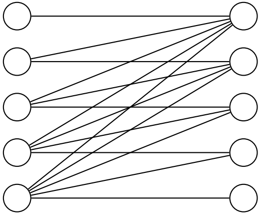


Stable and NIP for hereditary classes

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

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

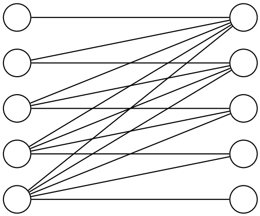
ladder



Stable and NIP for hereditary classes

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

ladder

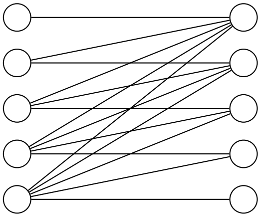


Bounded-degree graphs → stable
Unit interval graphs → NIP but not stable
Interval graphs → not NIP

Stable and NIP for hereditary classes

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

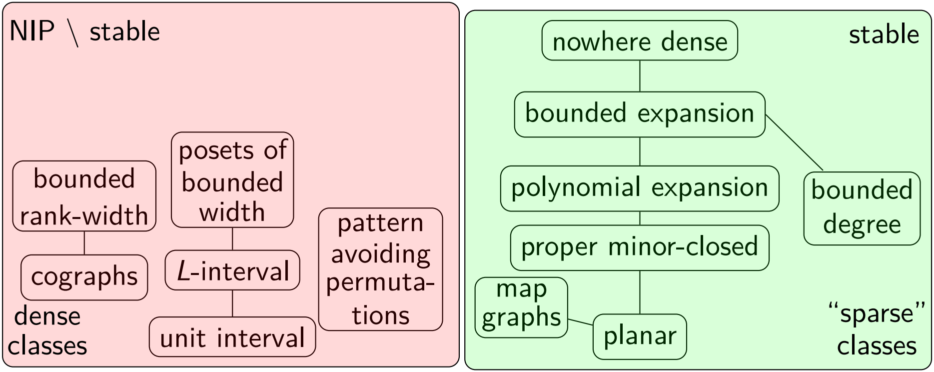
ladder



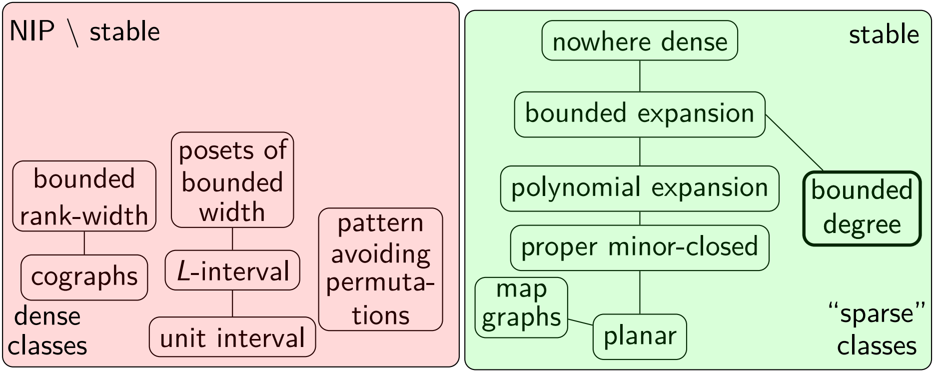
Bounded-degree graphs → stable
Unit interval graphs → NIP but not stable
Interval graphs → not NIP

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

Classes with known tractable FO model checking

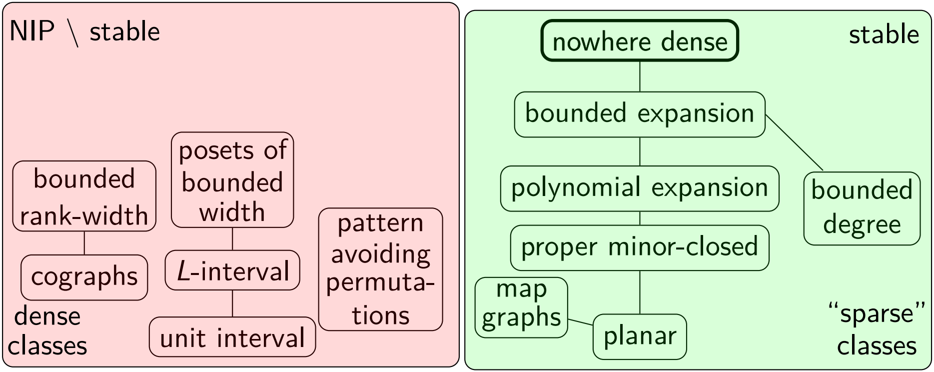


Classes with known tractable FO model checking



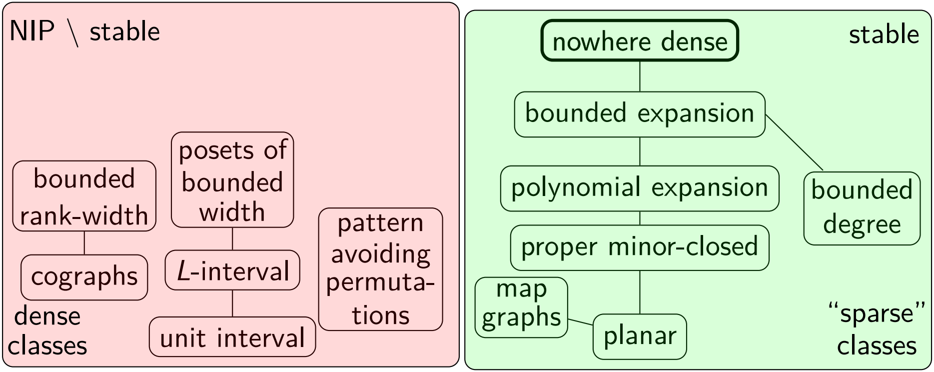
FO Model Checking solvable in $f(|\varphi|)n$ on bounded-degree graphs
[Seese '96]

Classes with known tractable FO model checking



FO Model Checking solvable in $f(|\varphi|)n^{1+\varepsilon}$ on any nowhere dense class [Grohe, Kreutzer, Siebertz '14]

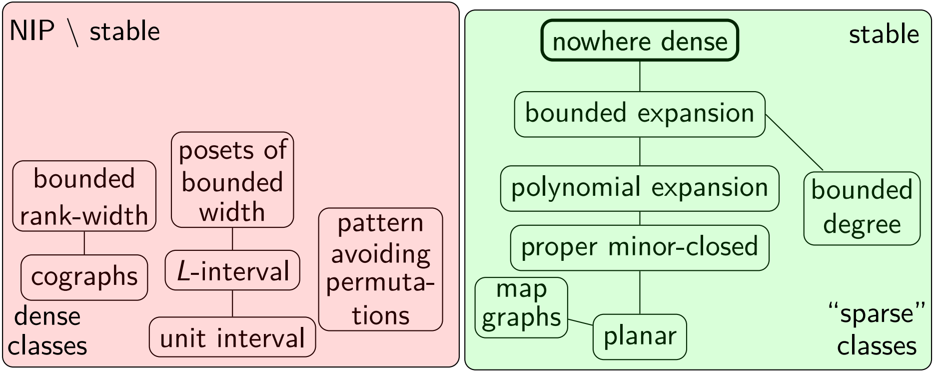
Classes with known tractable FO model checking



End of the story for the subgraph-closed classes

tractable FO MODEL CHECKING $\Leftrightarrow$ nowhere dense $\Leftrightarrow$ stable

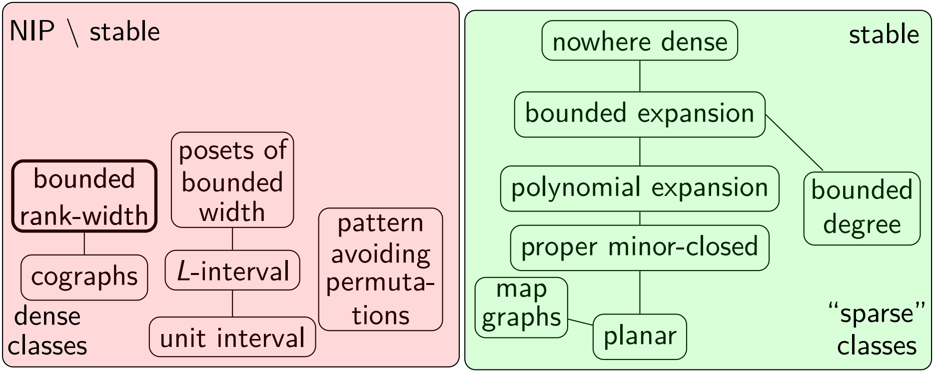
Classes with known tractable FO model checking



New program: transductions of nowhere dense classes

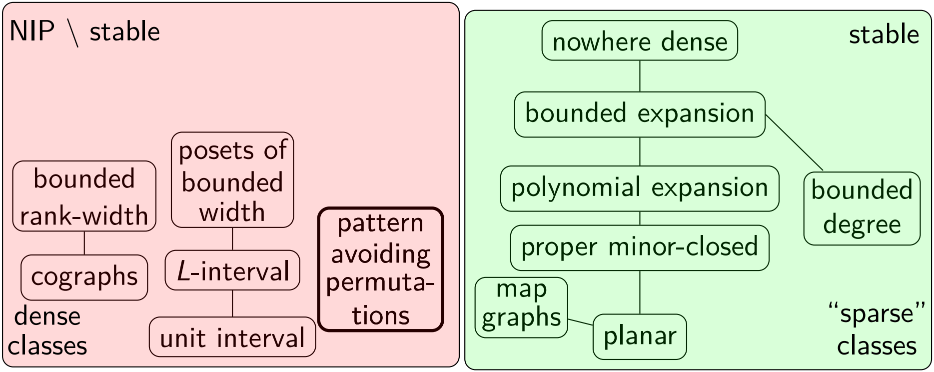
Not sparse anymore but still stable

Classes with known tractable FO model checking



MSO$_1$ Model Checking solvable in $f(|\varphi|, w)n$ on graphs of rank-width w
[Courcelle, Makowsky, Rotics '00]

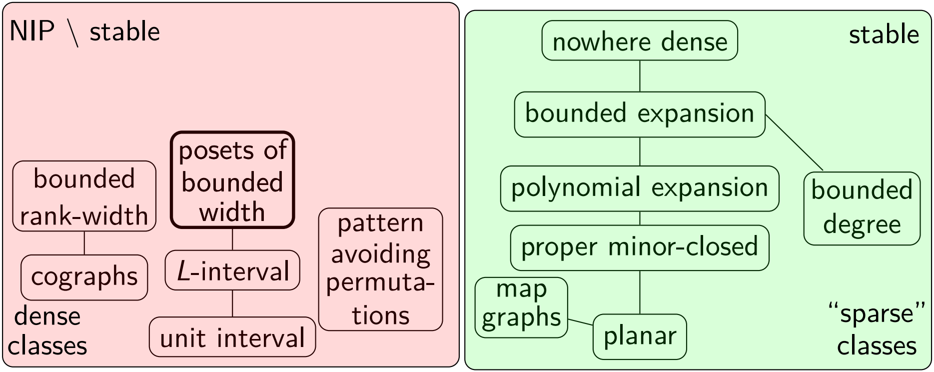
Classes with known tractable FO model checking



Is σ a subpermutation of τ? solvable in $f(|\sigma|)|\tau|$

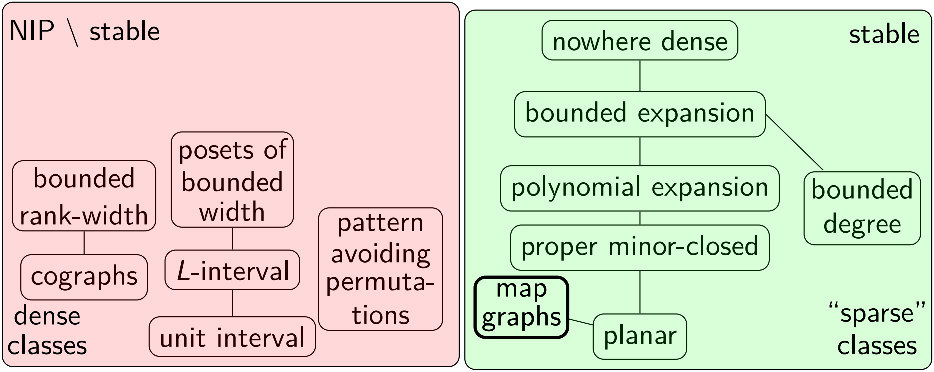
[Guillemot, Marx '14]

Classes with known tractable FO model checking



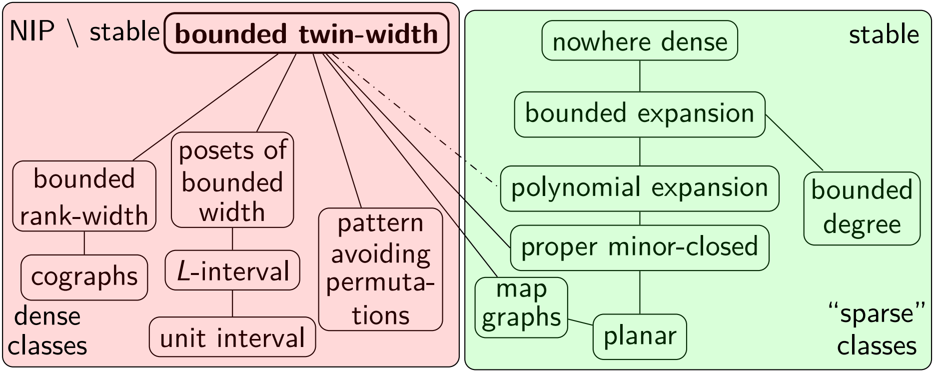
FO Model Checking solvable in $f(|\varphi|, w)n^2$ on posets of width w [GHLOORS '15]

Classes with known tractable FO model checking



FO MODEL CHECKING solvable in $f(|\varphi|)n^{O(1)}$ on map graphs
[Eickmeyer, Kawarabayashi '17]

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 partitioned local 1-type

Linearly 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 monadically dependent.*

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

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

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

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

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

Stable and structurally sparse classes

Conjecture (Ossona de Mendez)

Every monadically stable class is the FO transduction of a nowhere dense class.

Morally: *Stability coincides with structural sparsity*

Stable and structurally sparse classes

Conjecture (Ossona de Mendez)

Every monadically stable class is the FO transduction of a nowhere dense class.

Shown among classes of bounded linear cliquewidth, cliquewidth, and now twin-width:

Theorem (Gajarský, Pilipczuk, Toruńczyk '22)

Every stable class of bounded twin-width is the FO transduction of a class of bounded twin-width without arbitrarily large bicliques.

Stable and structurally sparse classes

Conjecture (Ossona de Mendez)

Every monadically stable class is the FO transduction of a nowhere dense class.

Shown among classes of bounded linear cliquewidth, cliquewidth, and now twin-width:

Theorem (Gajarský, Pilipczuk, Toruńczyk '22, Tww II '21)

Every stable class of bounded twin-width is the FO transduction of a class of bounded expansion.

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**

Compiling bounded twin-width graphs as p-f permutations

Our next goal:

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.

Compiling bounded twin-width graphs as p-f permutations

Our next goal:

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

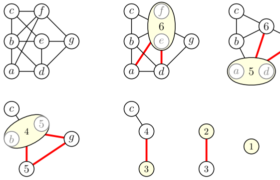
A class of binary structures has bounded twin-width if and only if it is a first-order transduction of a proper permutation class.

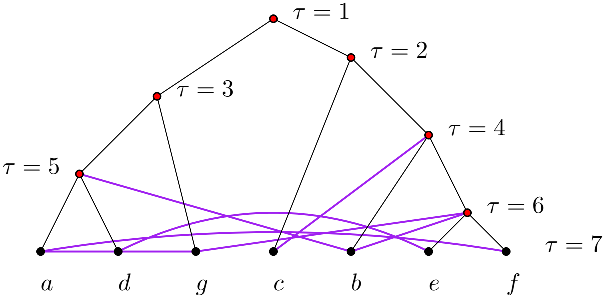
"if direction:" proper permutation classes have bounded twin-width + FO transductions preserve bounded twin-width

We now want to show:

$\forall$ class $\mathcal{C}$ of bounded twin-width, $\exists$ permutation class $\mathcal{P}$ avoiding one permutation and an FO transduction $\mathscr{T}$ such that $\mathcal{C} \subseteq \mathscr{T}(\mathcal{P})$.

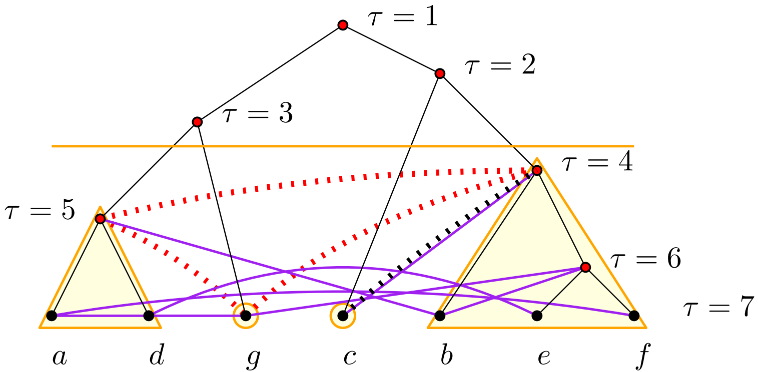
Twin-decomposition



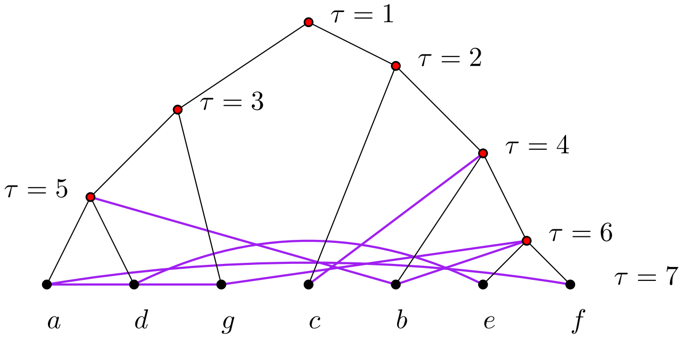


Contraction tree + transversal adjacencies (bicliques) + time τ

Reading out trigraphs from a twin-decomposition



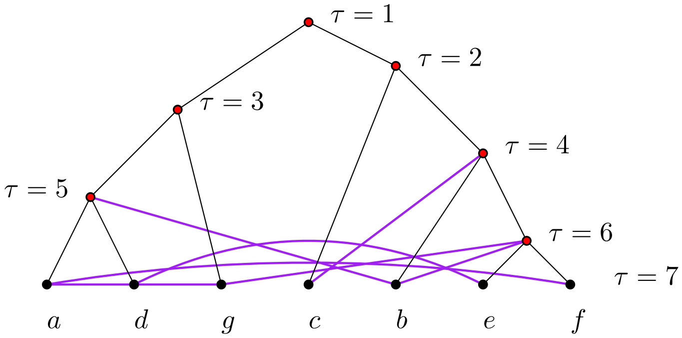
Twin-models



Twin-model: tree edges T, transversal edges V

Example: $T(3,5)$, $V(4,c)$

Twin-models

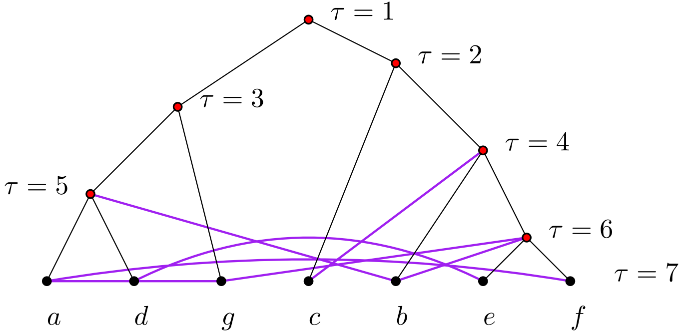


Twin-model: tree edges T, transversal edges V

Full twin-model: ancestor–descendant relation $\prec$, V

Example: $2 \prec e$

Twin-models



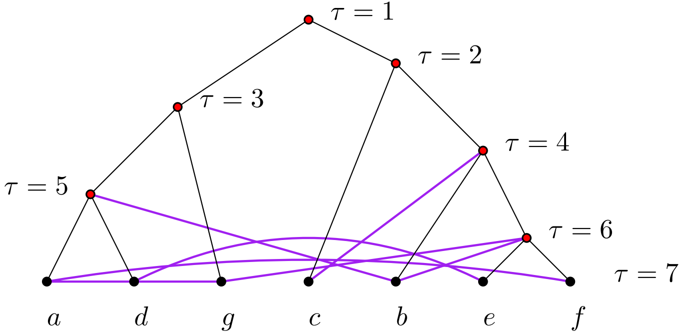
Twin-model: tree edges T, transversal edges V

Full twin-model: ancestor–descendant relation $\prec$, V

Ordered twin-model: T, tree pre-order $<$, V

$1 < 3 < 5 < a < d < g < 2 < c < 4 < b < 6 < e < f$

Twin-models

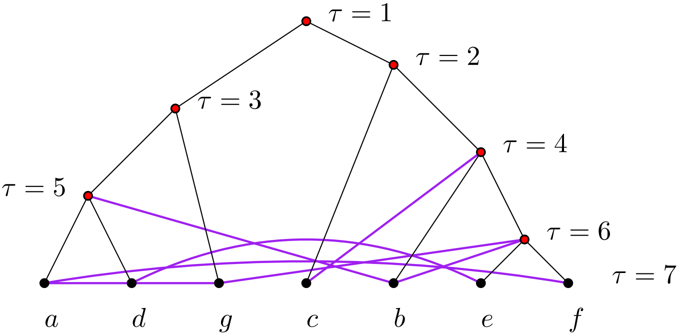


Twin-model: tree edges T, transversal edges V

Full twin-model: ancestor–descendant relation $\prec$, V

Ordered twin-model: T, tree pre-order $<$, V

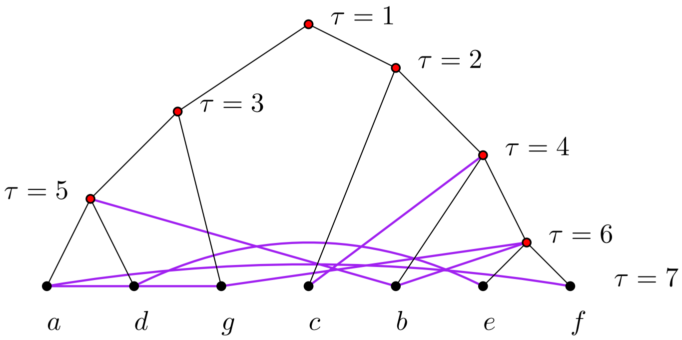
Why full twin-models?



One can FO reconstruct the initial graph from a full twin-model

$$E(x, y) := \exists x' \exists y' \left(x' \preceq x \ \wedge \ y' \preceq y \ \wedge \ V(x', y') \right)$$

Why full twin-models?

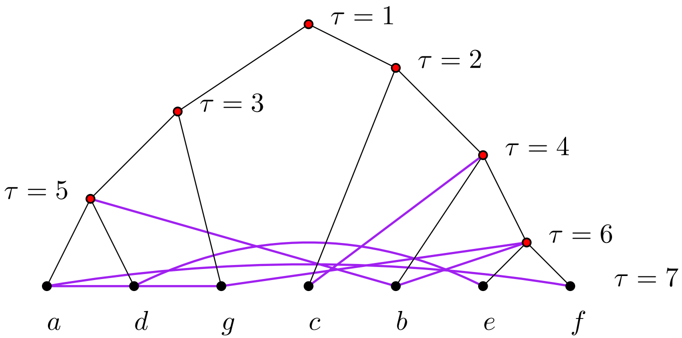


One can FO reconstruct the initial graph from a full twin-model

$$E(x, y) := \exists x' \exists y' \left(x' \preceq x \ \wedge \ y' \preceq y \ \wedge \ V(x', y')\right)$$

Example: $E(c, f)$ *since* $c \preceq c$, $4 \preceq f$, $V(4, c)$

Why full twin-models?

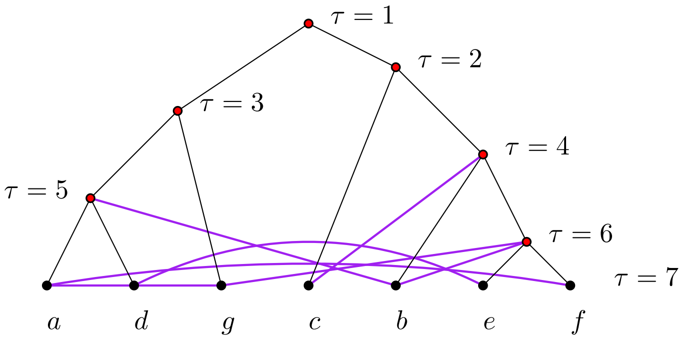


One can FO reconstruct the initial graph from a full twin-model

$$E(x, y) := \exists x' \exists y' \left(x' \preceq x \ \wedge \ y' \preceq y \ \wedge \ V(x', y') \right)$$

but *not* from a mere twin-model, in general

Why ordered twin-models?

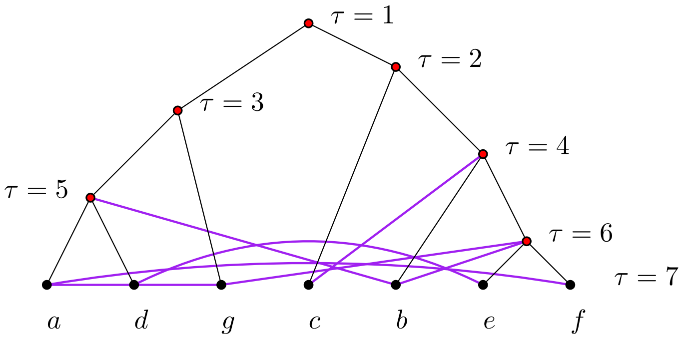


A linear order

$1 < 3 < 5 < a < d < g < 2 < c < 4 < b < 6 < e < f$

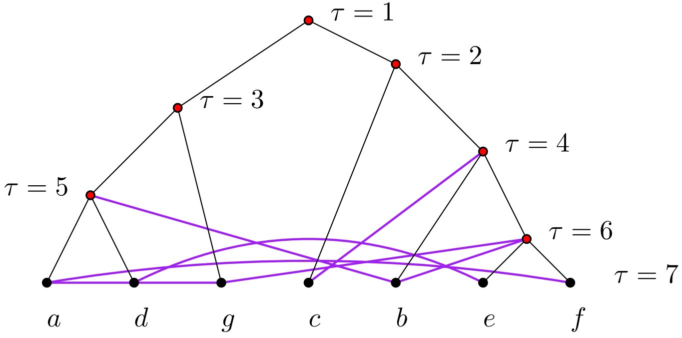
brings us closer to a permutation ($\equiv$ two linear orders)

Full and ordered twin-models are transduction equivalent



$$x \prec y := x < y \ \wedge \ \forall x < z \leq y \ \forall w \ T(z, w) \rightarrow x \leq w$$

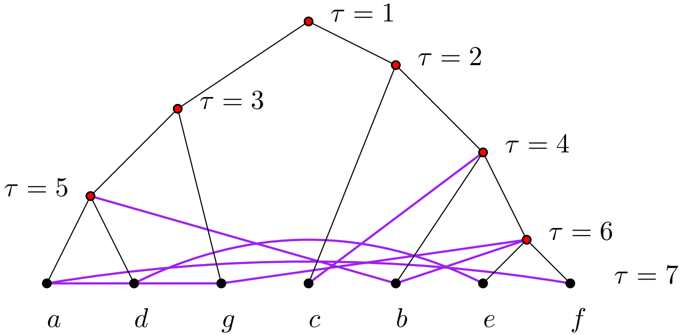
Full and ordered twin-models are transduction equivalent



$$x \prec y := x < y \ \wedge \ \forall x < z \leq y \ \forall w \ T(z, w) \rightarrow x \leq w$$

y is a strict descendant of x if it comes after in the pre-order, and every neighbor w (in the tree) of any intermediate z (possibly y) comes (non-strictly) after x

Full and ordered twin-models are transduction equivalent



To define $x < y$ from $\prec$, **mark each left child with one color**, , and express that the before-last vertex on the path from x to the least ancestor of x and y is marked (or simply $x \prec y$)

Done and left to do

$$\text{graphs} \twoheadleftarrow\!\!\longrightarrow \text{full twin-models} \twoheadleftarrow\!\!\longrightarrow\!\!\twoheadrightarrow \text{ordered twin-models}$$

bounded twin-width

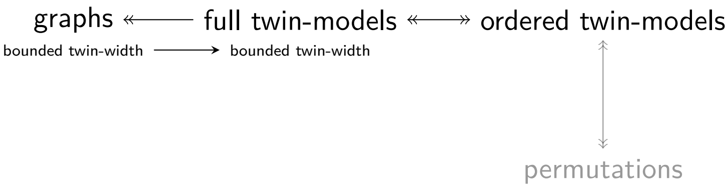
Done and left to do

graphs $\twoheadleftarrow$ full twin-models $\twoheadleftarrow\!\twoheadrightarrow$ ordered twin-models

bounded twin-width $\longrightarrow$ bounded twin-width

Mimicking a good contraction sequence on a full twin-model yields a good contraction sequence

Done and left to do



Past this point *bounded twin-width* is preserved by the FO transductions, and we just need to show that:

ordered twin-models and permutations are transduction equivalent

Sparsity of the twin-model

Twin-models have bounded twin-width and degeneracy

Sparsity of the twin-model

Twin-models have bounded twin-width and degeneracy

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

Bounded twin-width and degeneracy $\Rightarrow$ bounded expansion.

Sparsity of the twin-model

Twin-models have bounded twin-width and degeneracy

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

Bounded twin-width and degeneracy $\Rightarrow$ bounded expansion.

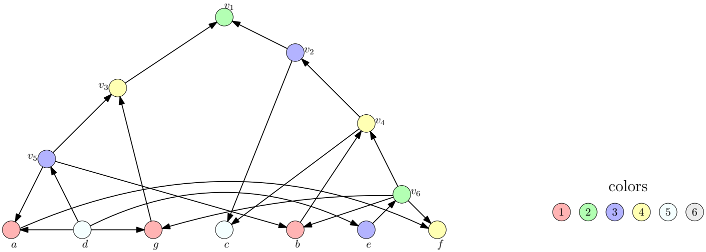
Theorem (Nešetřil, Ossona de Mendez '08)

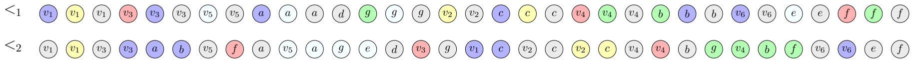
Bounded expansion $\Rightarrow$ bounded star chromatic number.

I.e., proper $O(1)$-coloring such that every two colors induce a disjoint union of stars

Encoding: Ordered twin-models to permutations

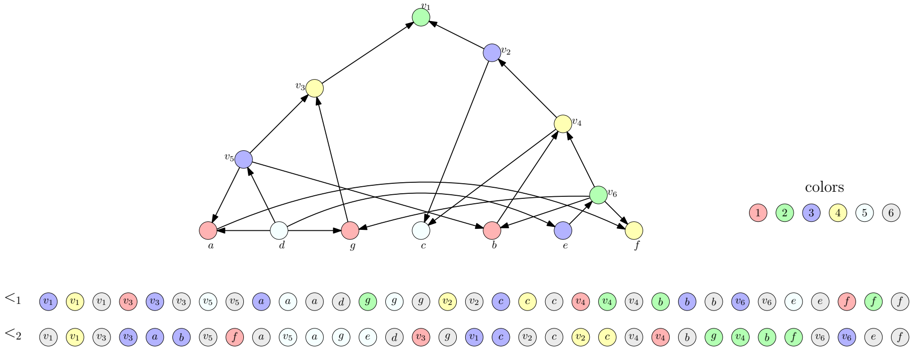
Fix a star coloring and orient edges away from centers of stars
$\rightarrow$ bounded in-degree





Encoding: Ordered twin-models to permutations

Fix a star coloring and orient edges away from centers of stars
$\rightarrow$ bounded in-degree



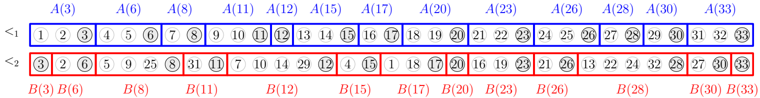
List in the pre-order traversal:

- $<_1$: the incoming arcs
- $<_2$: the outgoing arcs

where an arc is a copy of its out-vertex with color of its in-vertex

Decoding: Permutations to ordered twin-models

Guess the block ends (color 1)

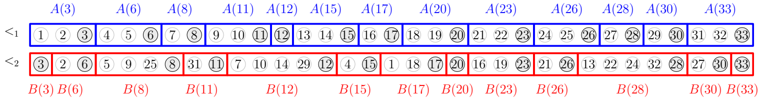


$3 < 6 < 8 < 11 < 12 < 15 < 17 < 20 < 23 < 26 < 28 < 30 < 33$

is the tree pre-order (on the domain of the image)

Decoding: Permutations to ordered twin-models

Guess the block ends (color 1)



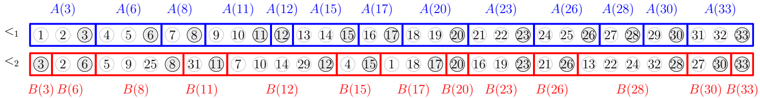
$3 < 6 < 8 < 11 < 12 < 15 < 17 < 20 < 23 < 26 < 28 < 30 < 33$

is the tree pre-order (on the domain of the image)

Two vertices are adjacent if their blocks along $<_1$ and $<_2$ contain a same element (namely, their linking arc)

Decoding: Permutations to ordered twin-models

Guess the block ends (color 1)



$3 < 6 < 8 < 11 < 12 < 15 < 17 < 20 < 23 < 26 < 28 < 30 < 33$

is the tree pre-order (on the domain of the image)

Two vertices are adjacent if their blocks along $<_1$ and $<_2$ contain a same element (namely, their linking arc)

Use an extra color for the transversal edges (color 2)

Recent developments

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.

Recent developments

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.

Recent developments

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.

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 $\mathcal{P}$.

The lens of contraction sequences

Class of bounded	FO transduction of	constr. on red graphs	efficient MC
linear rank-width	linear order	bd #edges	MSO
rank-width	tree order	bd component	**MSO**
twin-width	**proper perm. class**	bd degree	**FO**

The lens of contraction sequences

Class of bounded	FO transduction of	constr. on red graphs	efficient MC
linear rank-width	linear order	bd #edges	MSO
rank-width	tree order	bd component	**MSO**
twin-width	**proper perm. class**	bd degree	**FO**

Thank you for your attention!