FO model checking on graphs of bounded twin-width

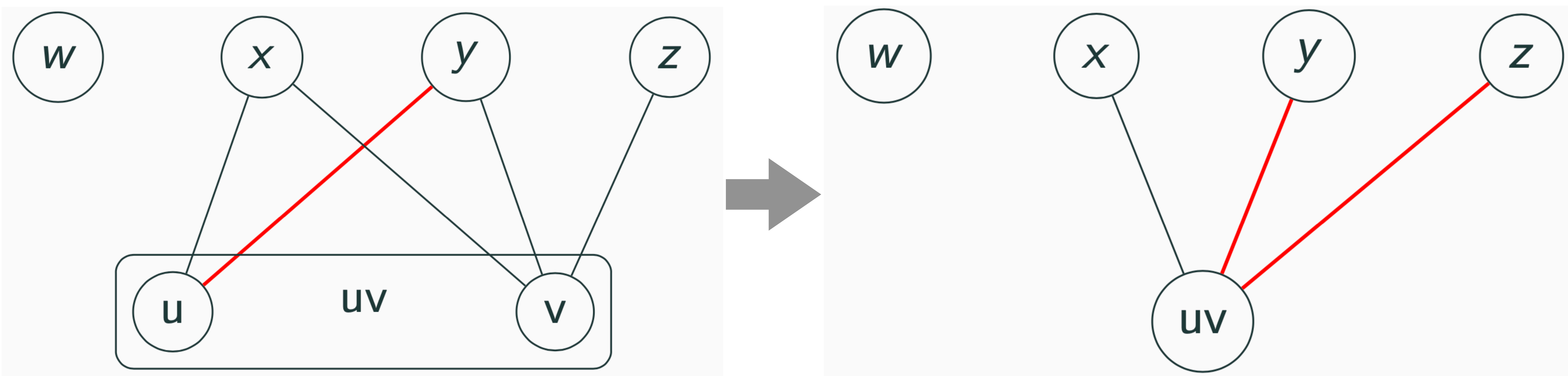
<u>Eunjung KIM</u>,
LAMSADE / CNRS, Université Paris-Dauphine

20èmes Journées de Combinatoire et d'Algorithmes du Littoral Méditerranéen
(JCALM'23)
13 December 2023, Montpellier, France

Contraction in a trigraph

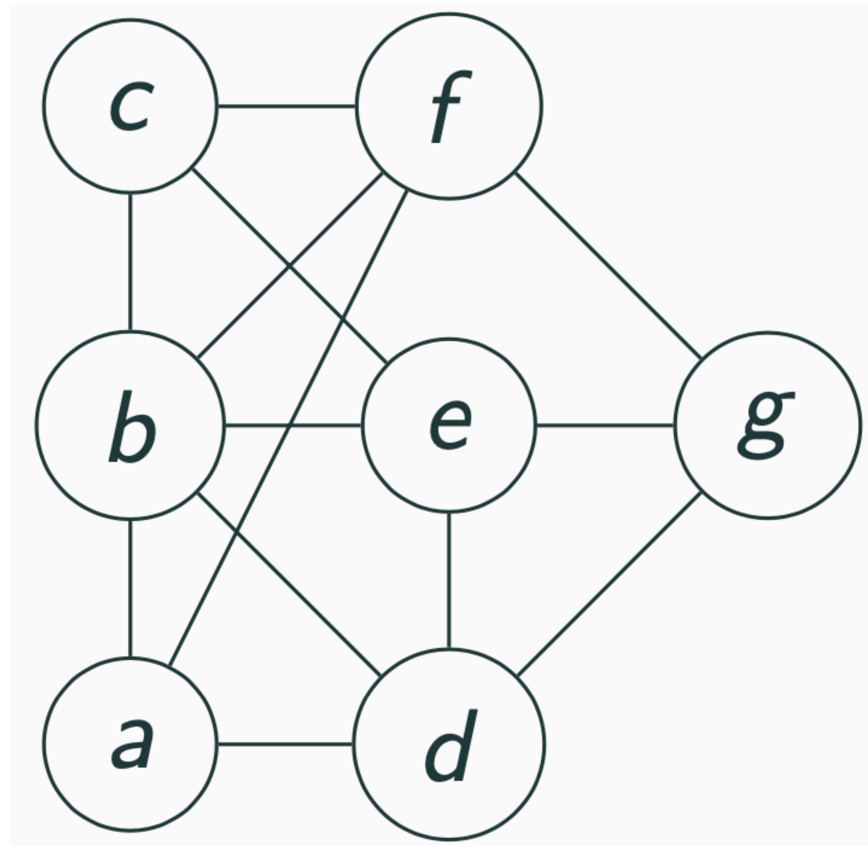
Trigraph has three types of adjacency: (black) edge, non-edge, red edge

Identification of two vertices, not-necessarily adjacent



- edges with $N(u) \triangle N(v)$ turn red
- red edges stay red

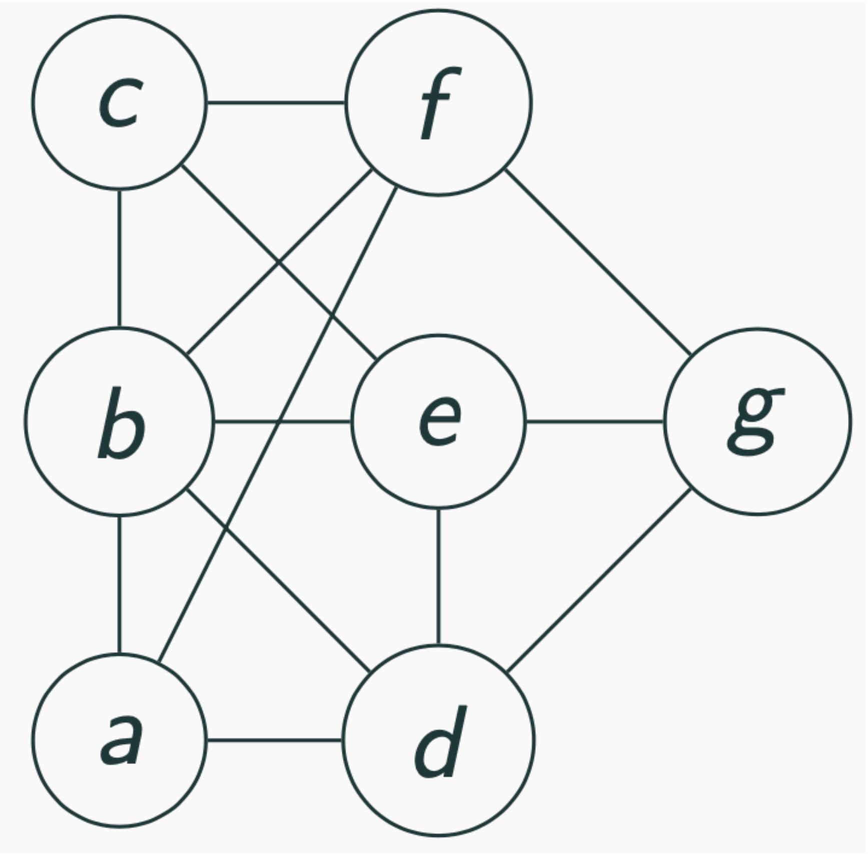
Contraction Sequence



A   contraction sequence of G =

a sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_1$= single-vertex graph
such that G_i is obtained from G_{i+1} by one contraction

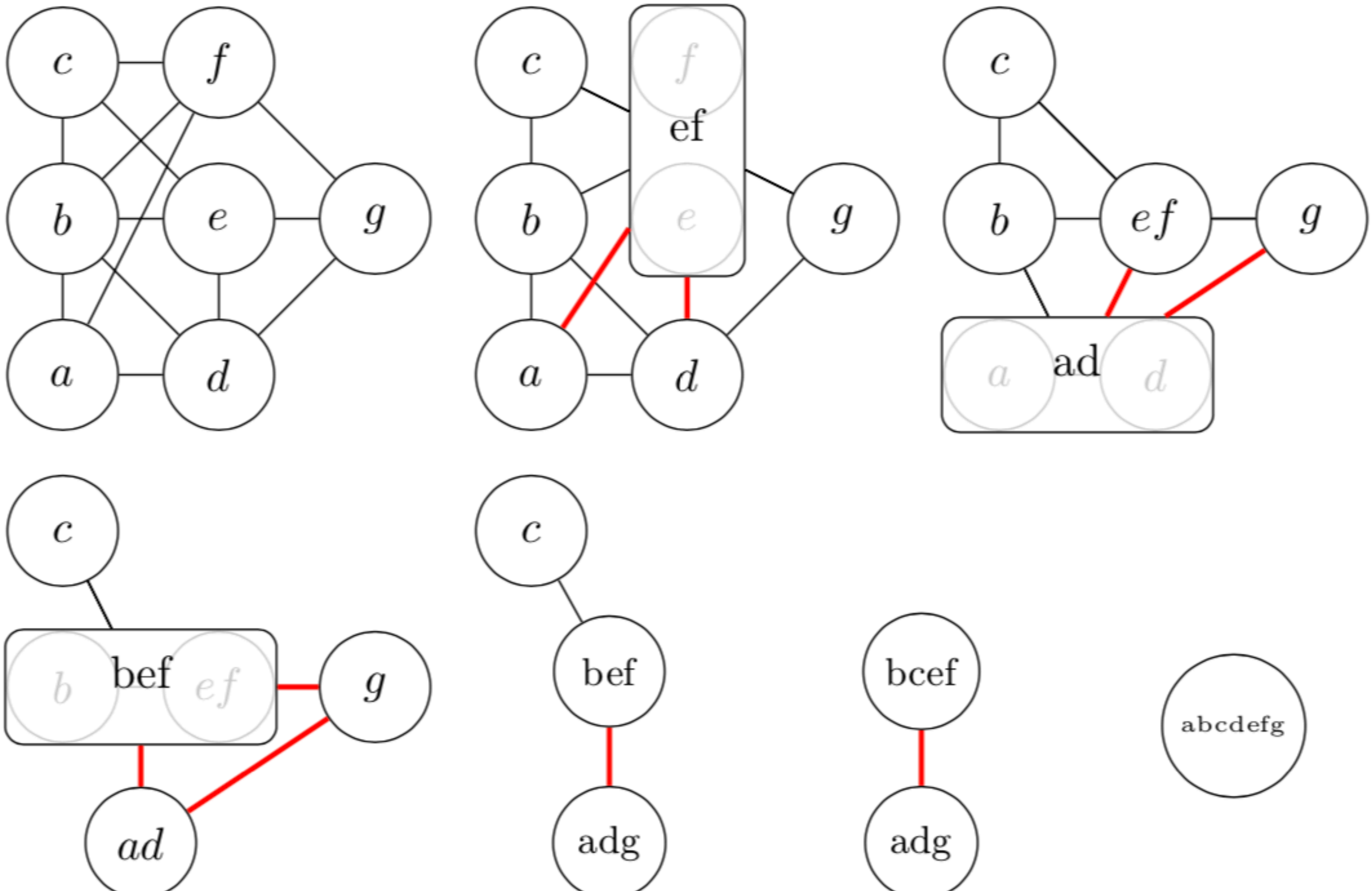
Contraction Sequence



A d-contraction sequence of G =

a sequence of trigraphs $G = G_n, G_{n-1}, \ldots, G_1$= single-vertex graph
such that G_i is obtained from G_{i+1} by one contraction
and the max red degree of each G_i is at most d.

2-contraction sequence



Twin-width of a graph

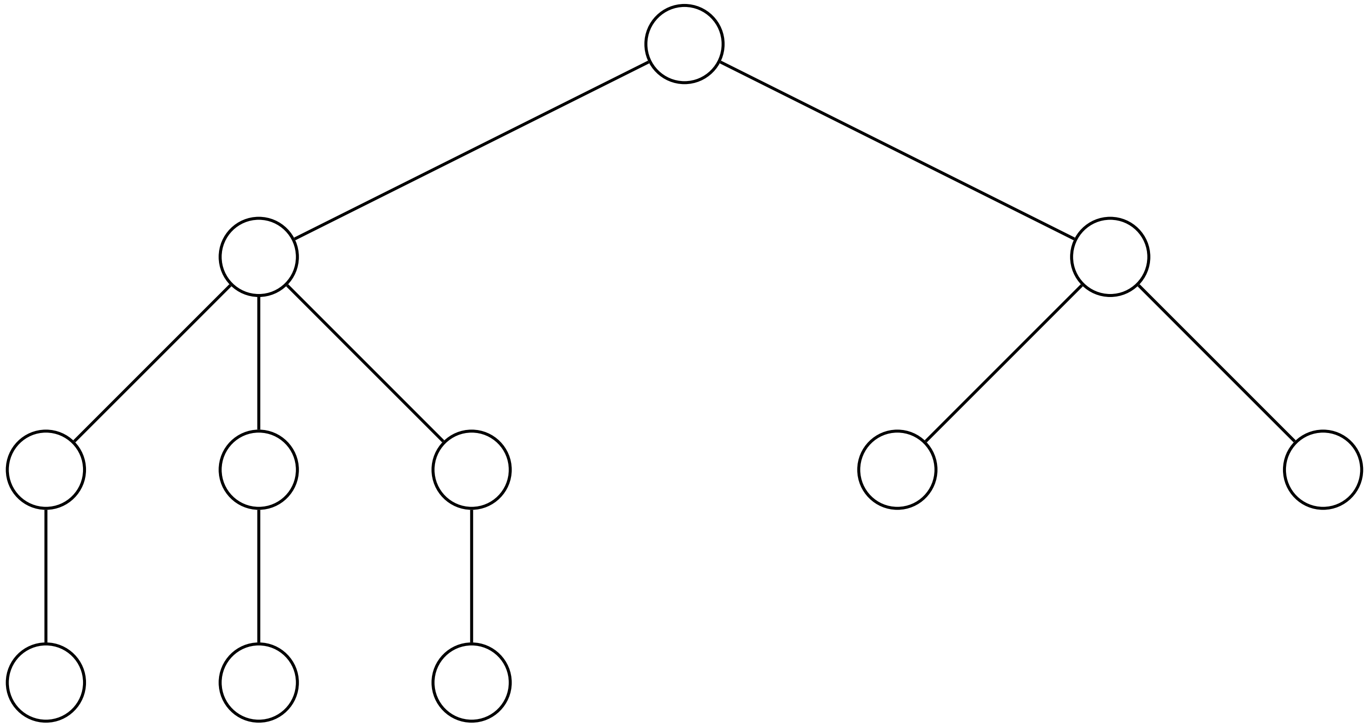
Twin-width of G =

the smallest d s.t. ∃ d-contraction sequence of G.

What is the (upper-bound of) twin-width of …

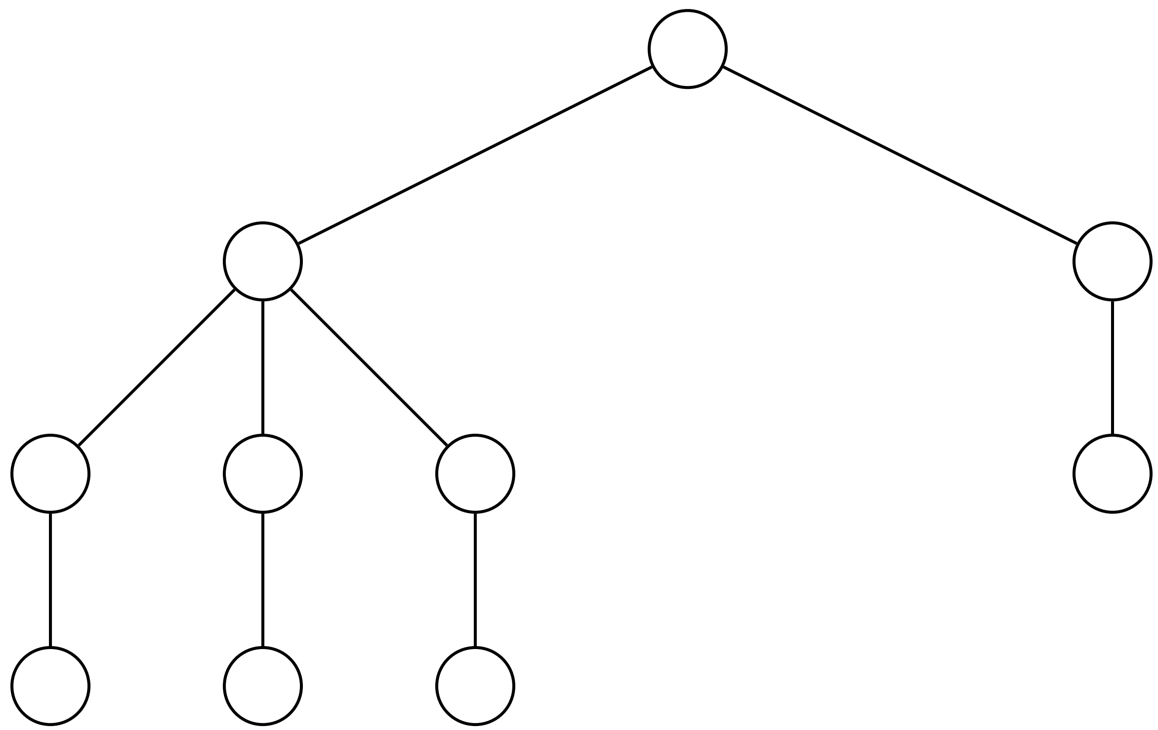
- clique?
- disjoint union of G and H?
- complete join of G and H?
- cograph?
- path?
- tree?
- planar graphs?

Trees



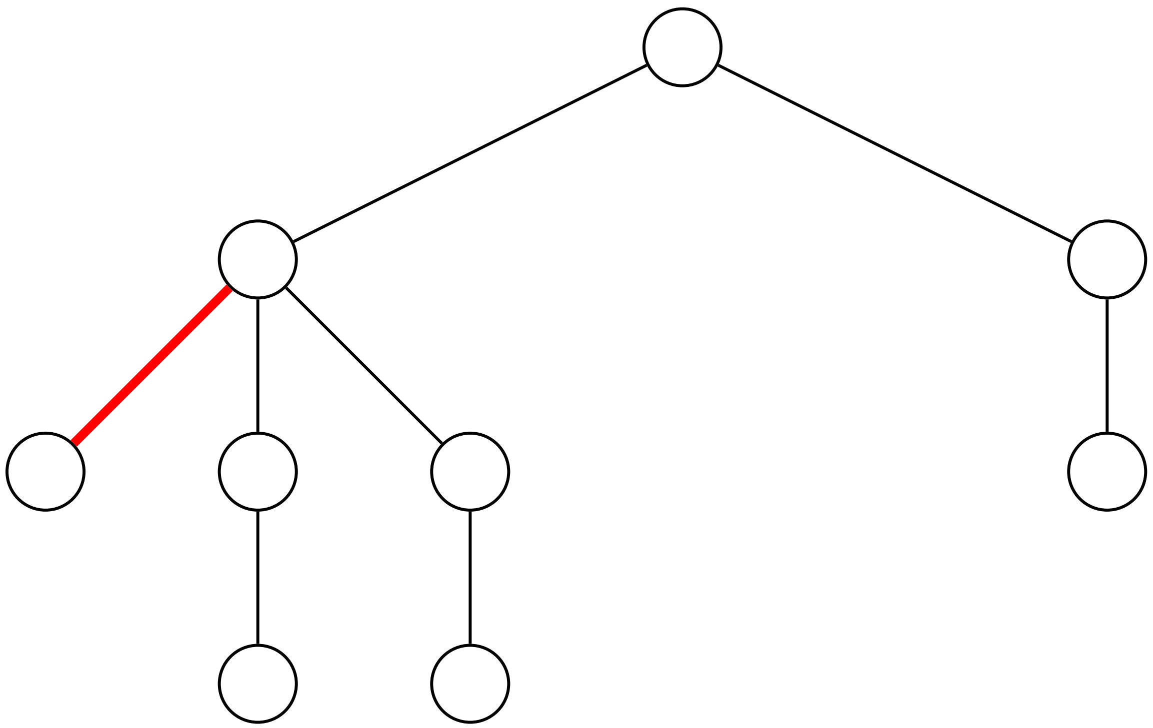
If possible, contract two twin leaves

Trees



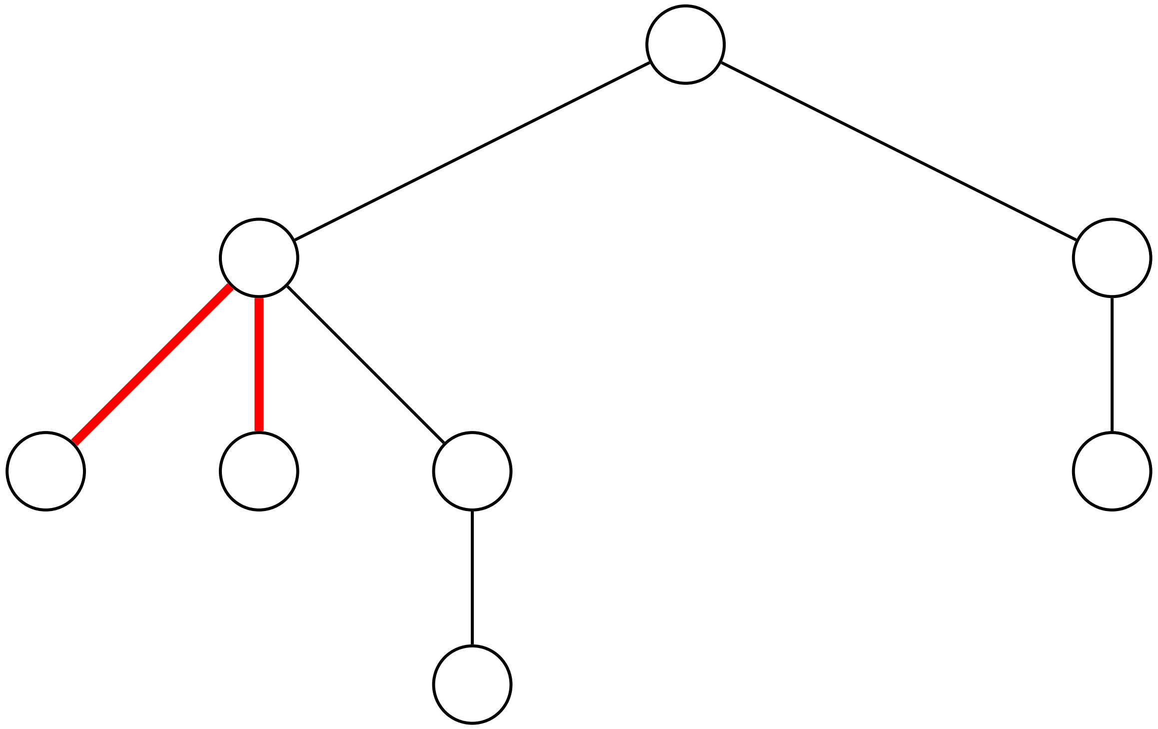
If not, contract a deepest leaf with its parent

Trees



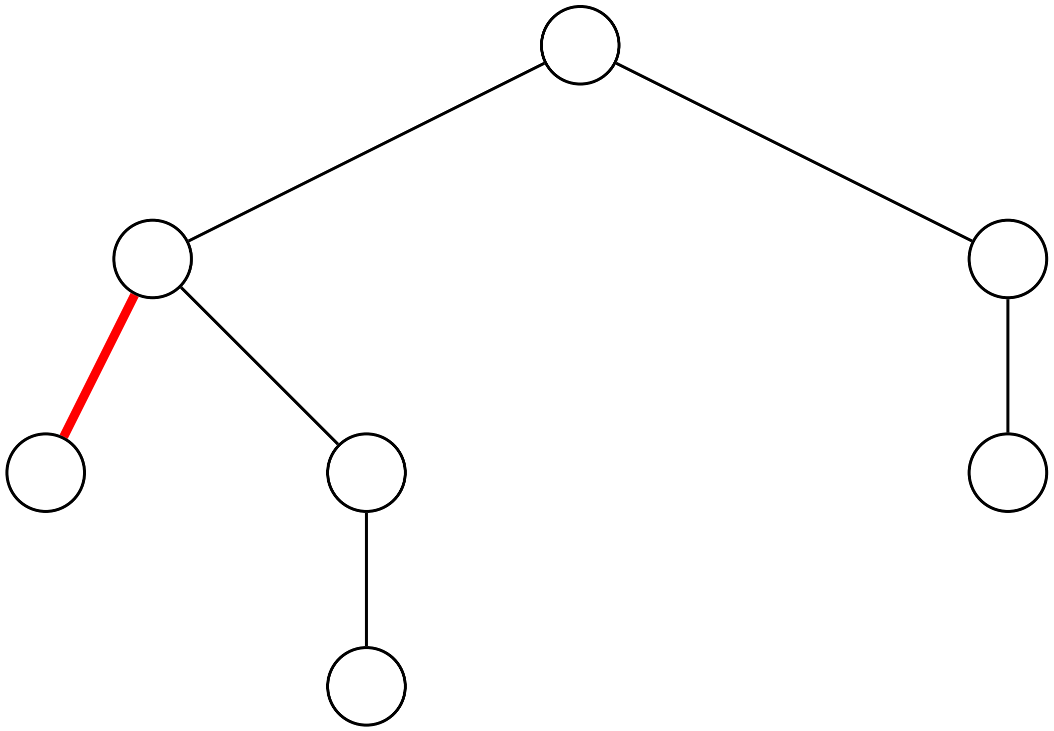
If not, contract a deepest leaf with its parent

Trees



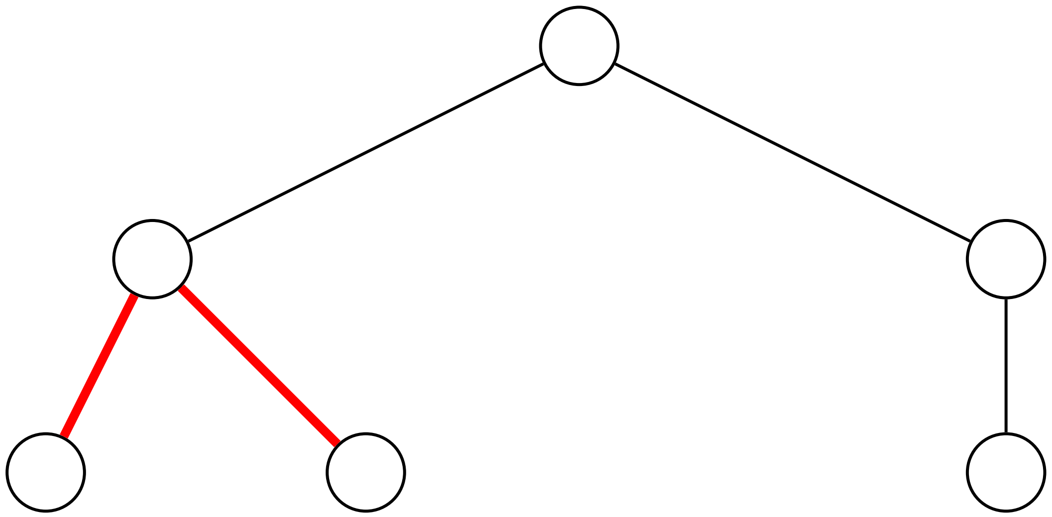
If possible, contract two twin leaves

Trees



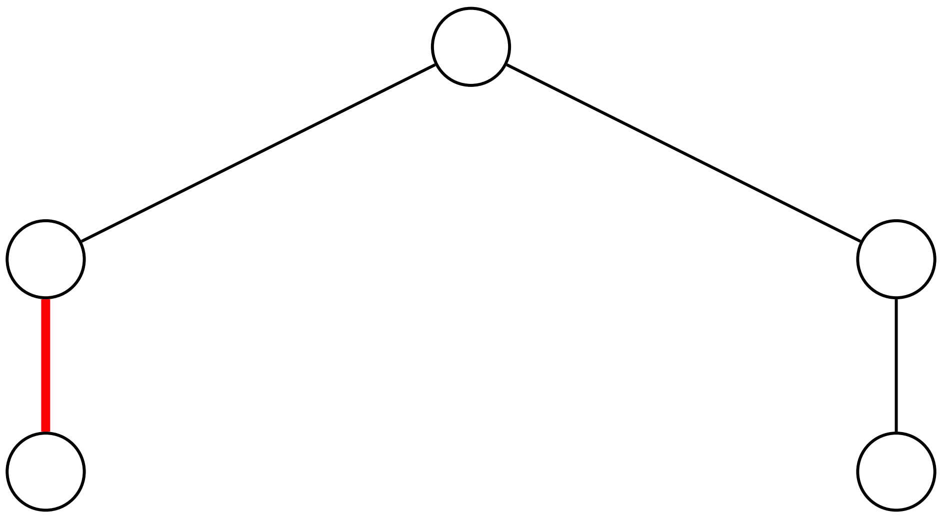
Cannot create a red degree-3 vertex

Trees



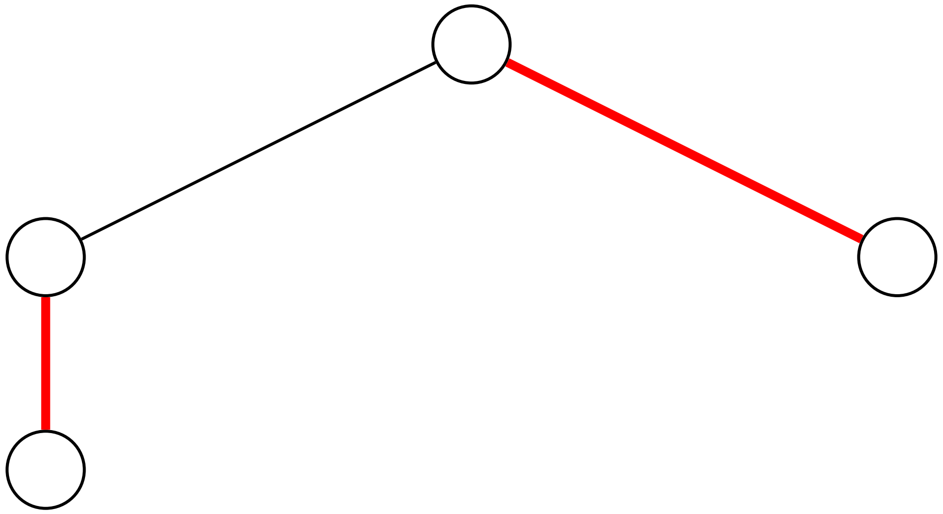
Cannot create a red degree-3 vertex

Trees



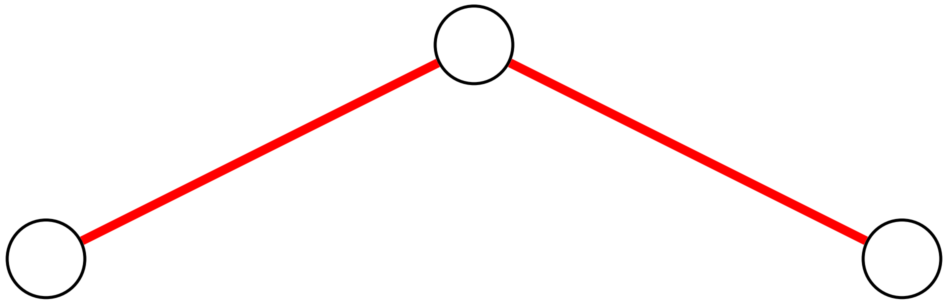
Cannot create a red degree-3 vertex

Trees



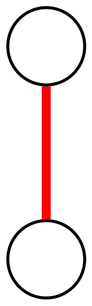
Cannot create a red degree-3 vertex

Trees



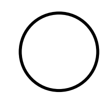
Cannot create a red degree-3 vertex

Trees



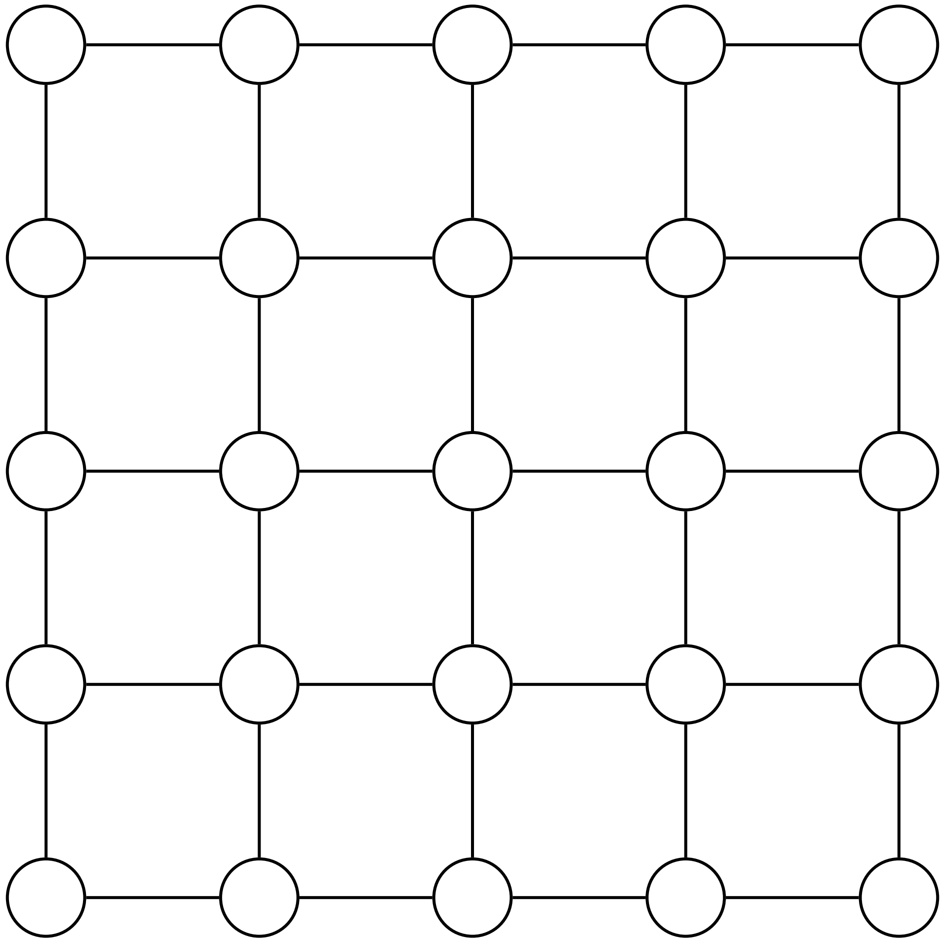
Cannot create a red degree-3 vertex

Trees

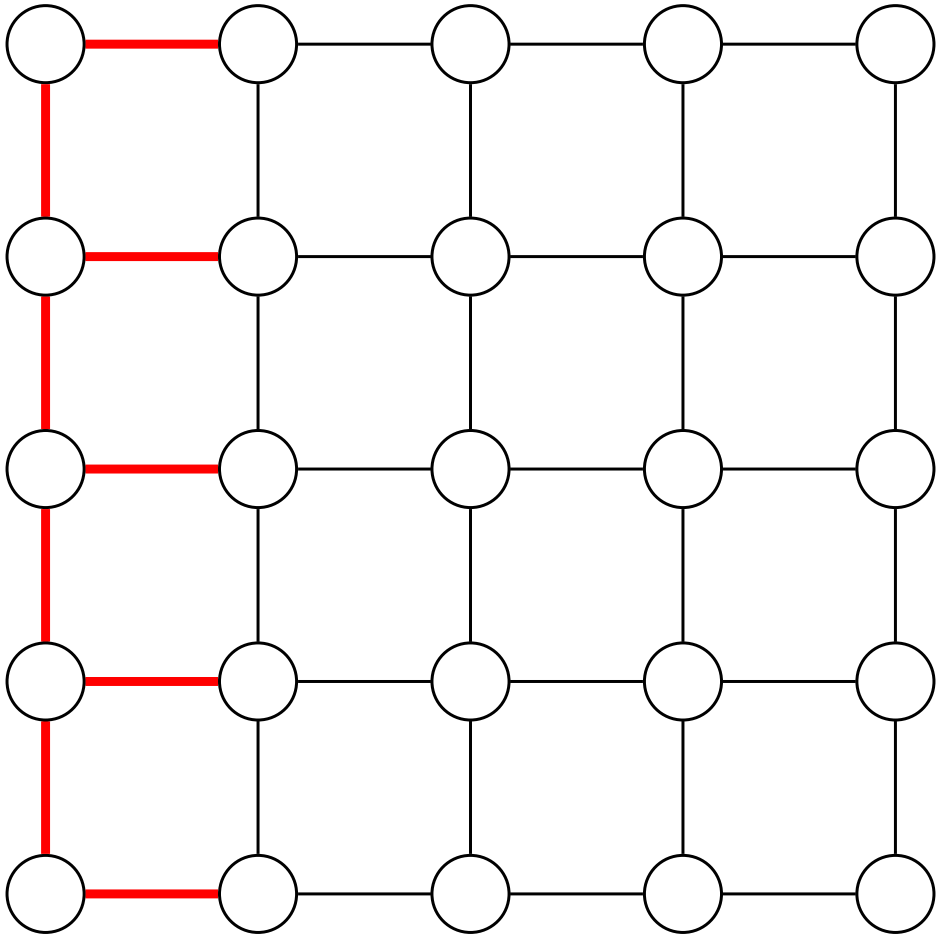


Generalization to bounded *treewidth* and even bounded *rank-width*

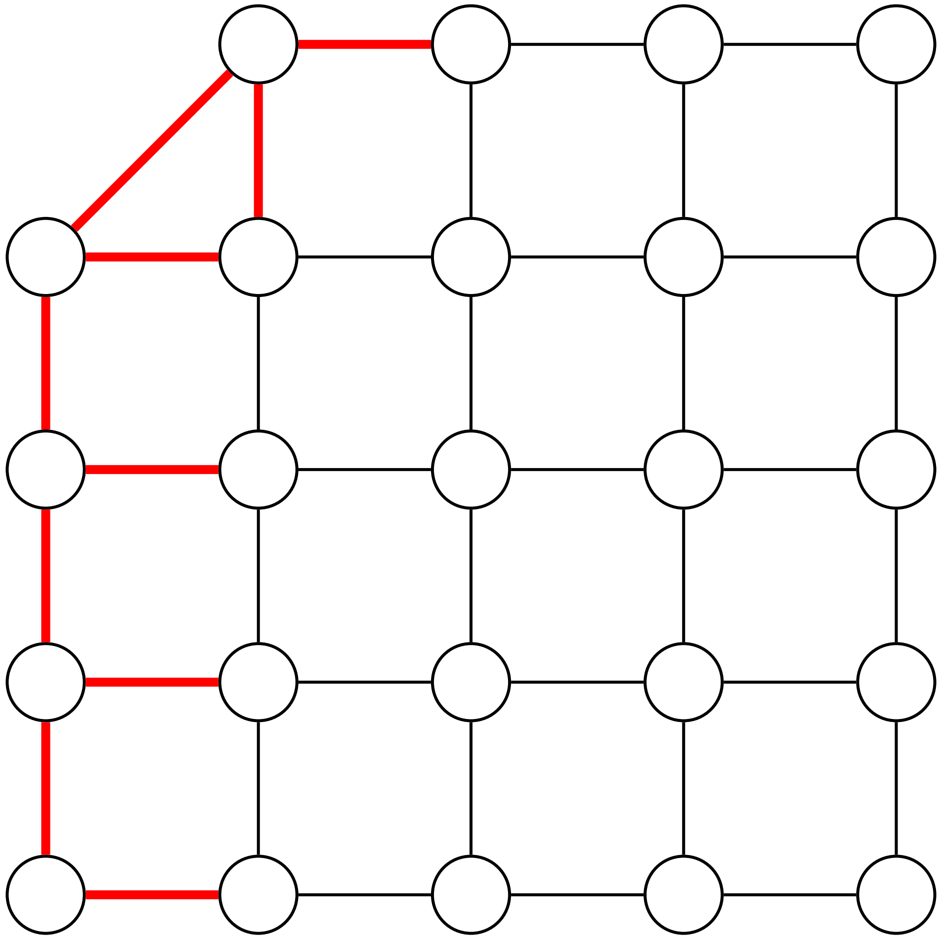
Grids



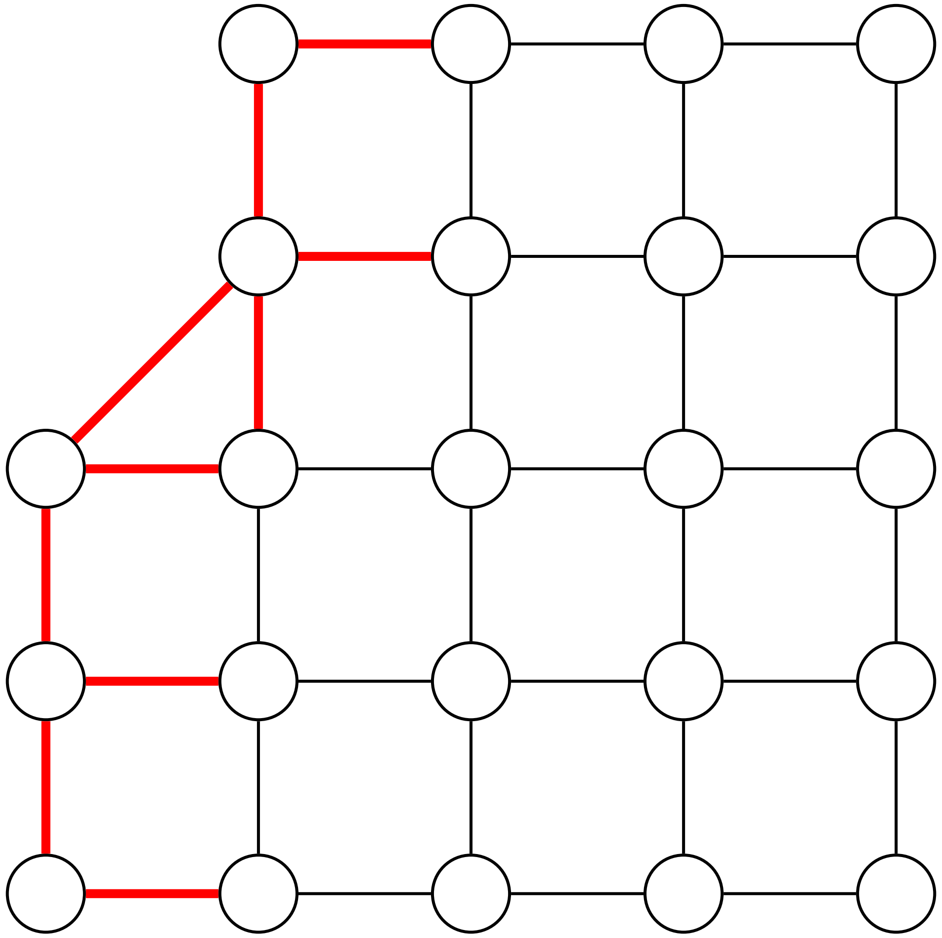
Grids



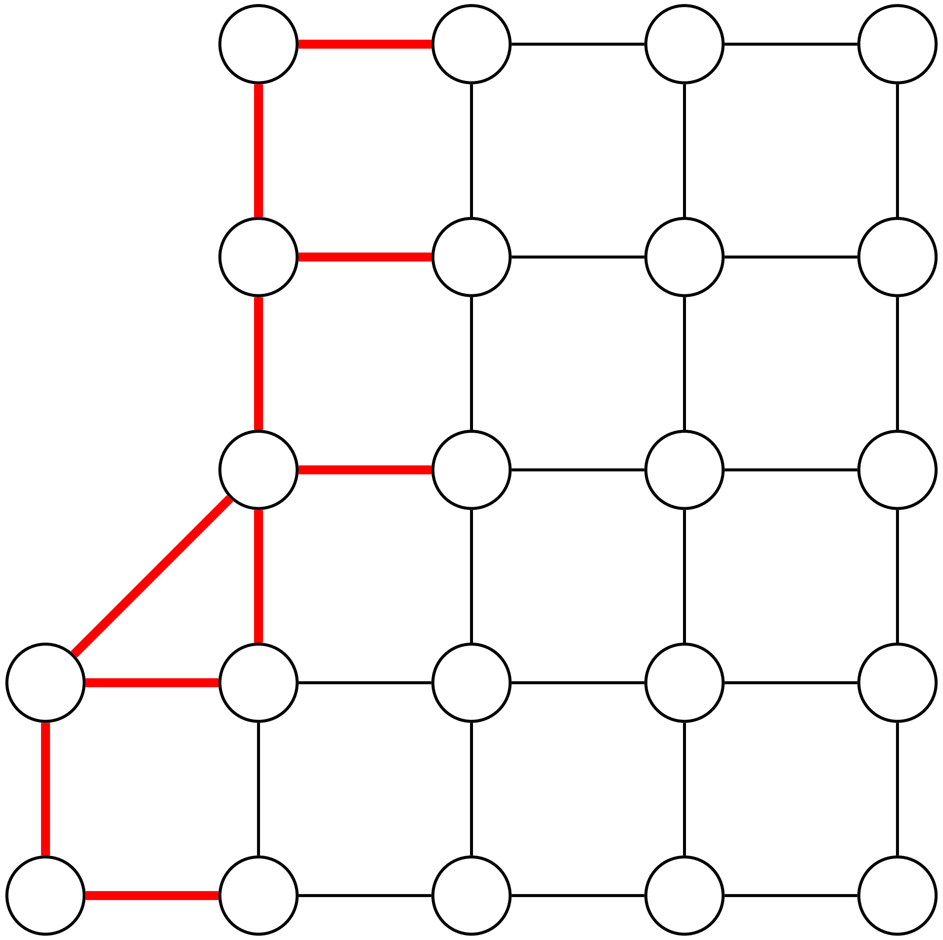
Grids



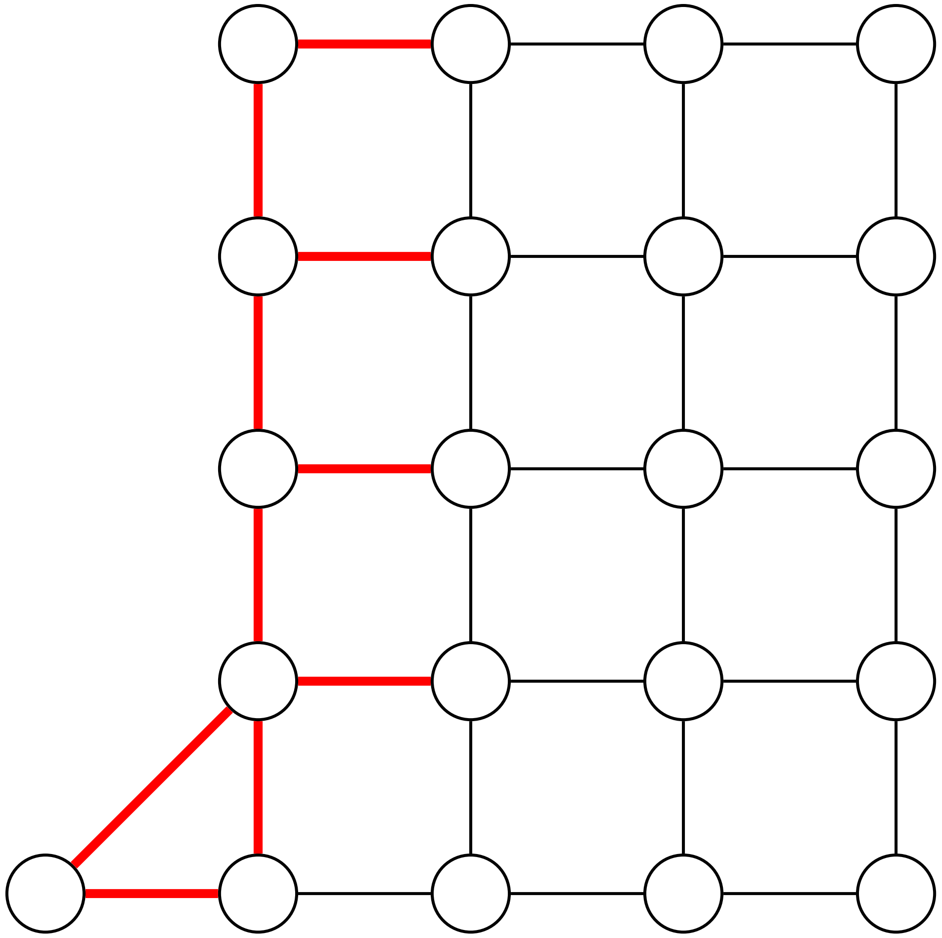
Grids



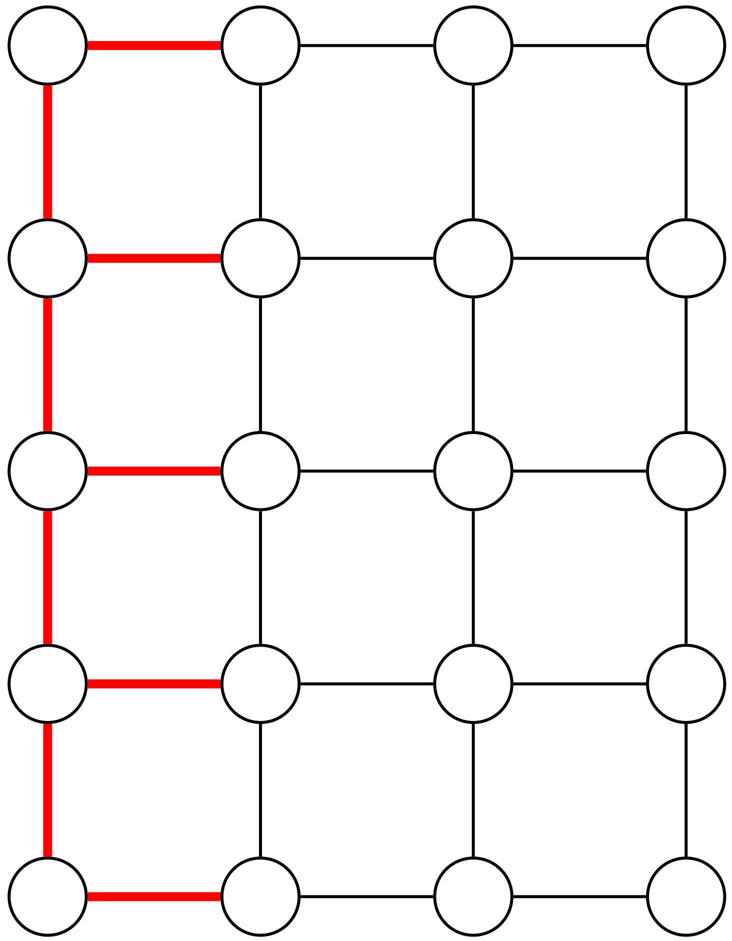
Grids



Grids



Grids



4-sequence for planar grids

Graph classes of small twin-width

[Bonnet, Geniat, K, Thomassé, Watrigant '20, '21]

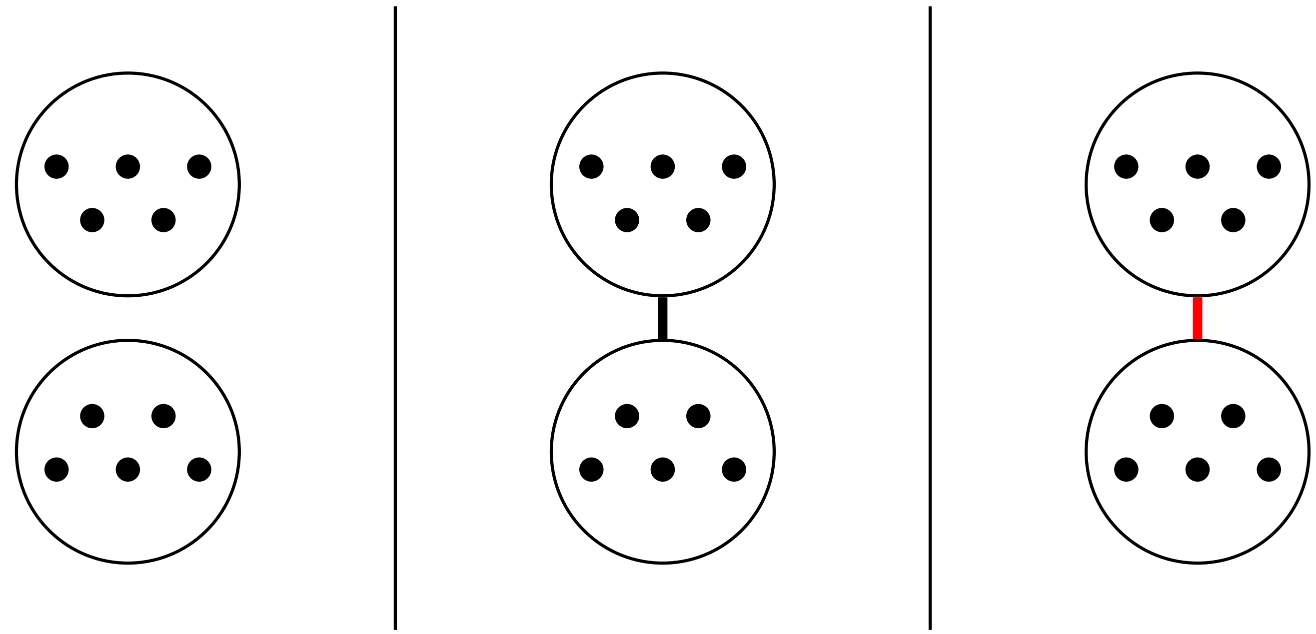
- trees, graphs of bounded tree-width
- bounded clique-width (rank-width) graphs
- unit interval graphs
- strong products of two graphs of bounded tww, one with bounded degree
- $\Omega(\log n)$-subdivision of all n-vertex graphs, etc.
- (subgraphs of) d-dimensional grids
- K_t-free unit ball graphs in dimension d
- hereditary proper subclass of permutation graphs
- posets of bounded antichain size
- K_t-minor-free graphs
- square of planar graphs
- map graphs
- k-planar graphs
- bounded degree string graphs

Graph classes of small twin-width

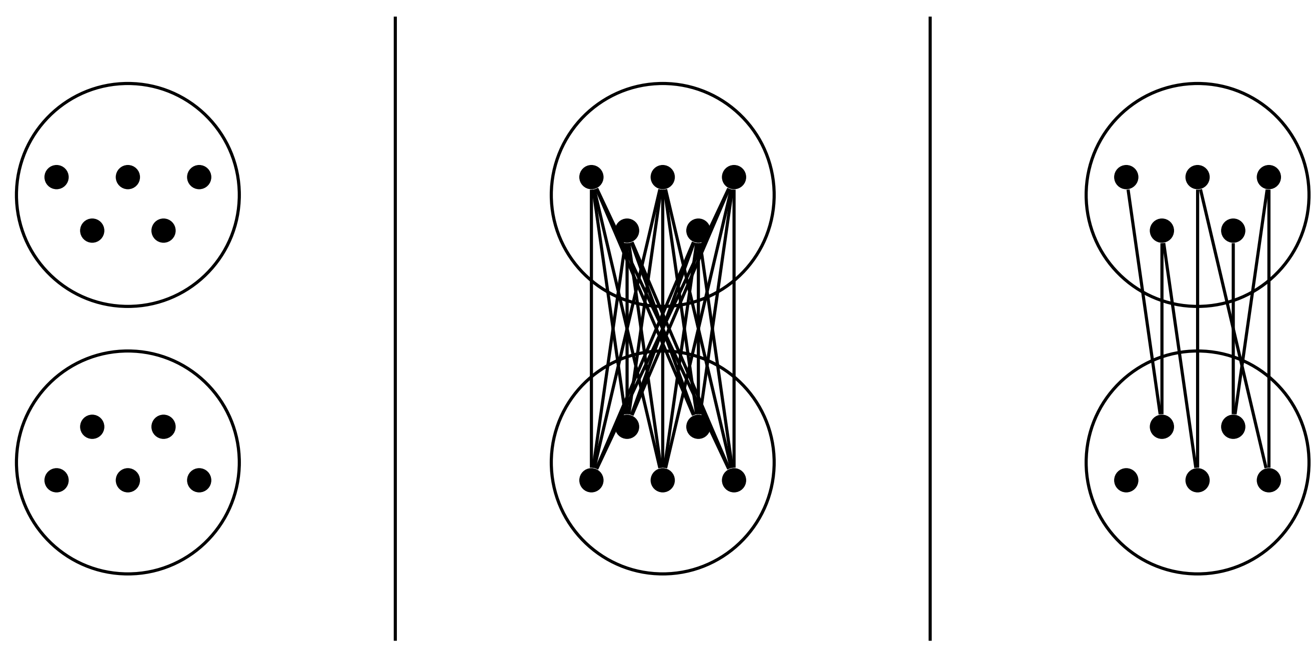
[Bonnet, Geniat, K, Thomassé, Watrigant '20, '21]

- trees, graphs of bounded tree-width
- bounded clique-width (rank-width) graphs
- unit interval graphs
- strong products of two graphs of bounded tww, one with bounded degree
- $\Omega(\log n)$-subdivision of all n-vertex graphs, etc.
- (subgraphs of) d-dimensional grids
- K_t-free unit ball graphs in dimension d
- hereditary proper subclass of permutation graphs
- posets of bounded antichain size
- K_t-minor-free graphs
- square of planar graphs
- map graphs
- k-planar graphs
- bounded degree string graphs

The class of all cubic graphs have unbounded twin-width given two bags:

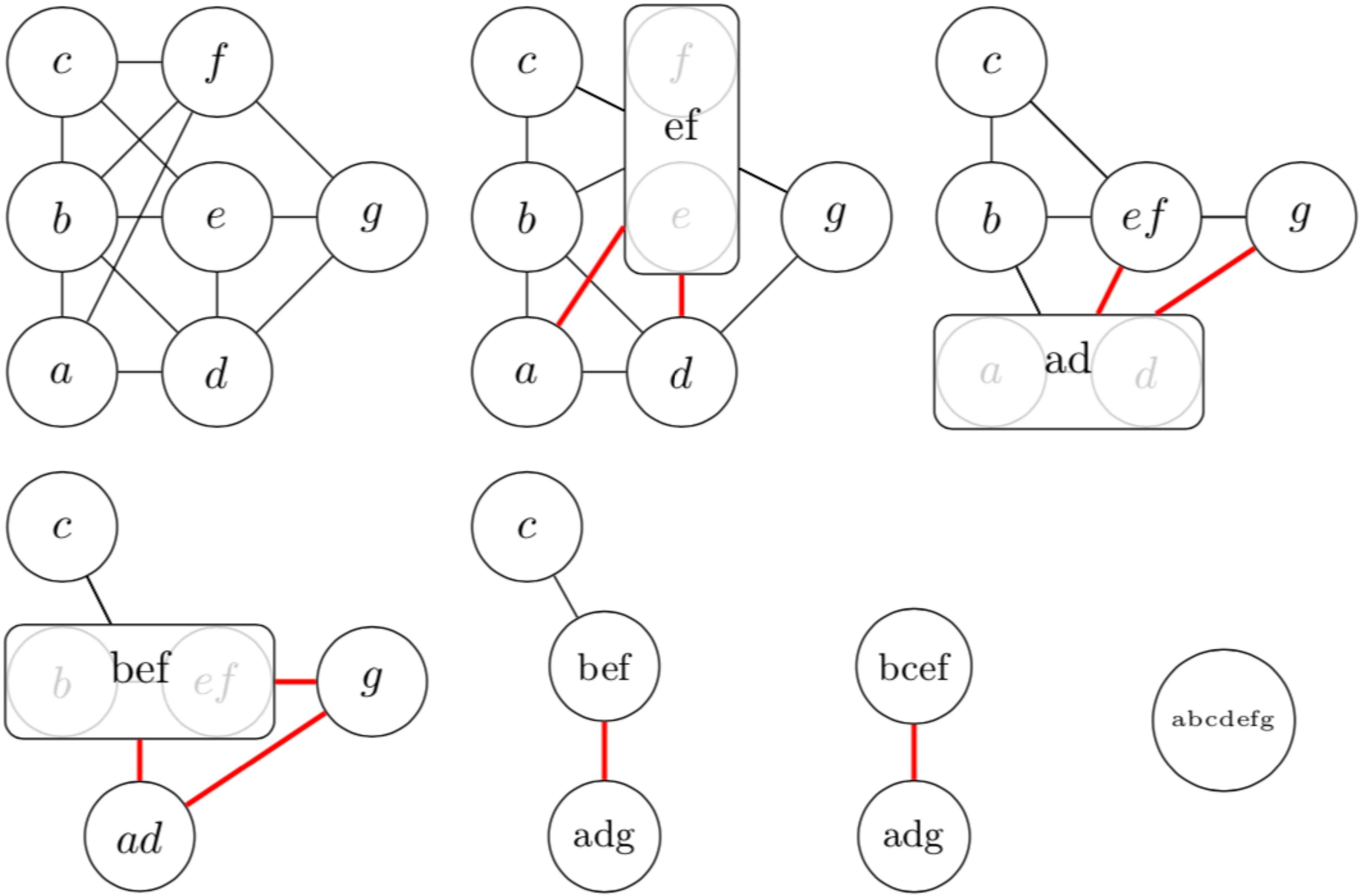


it means in the original graph:



no edge | all edges | at least one edge, at least one non-edge

2-partition sequence



Twin-width of a graph

A d-contraction sequence of G =

a sequence of partitions
$\mathscr{P}_n = \{\{v\} : v \in V(G)\}, \mathscr{P}_{n-1}, \ldots, \mathscr{P}_i, \ldots, \mathscr{P}_1 = \{V(G)\}$ such that $\mathscr{P}_i$ is obtained from P_{i+1} by merging two parts
and the max red degree of each quotient graph $G/\mathscr{P}_i$ is at most d.

Twin-width of G =

the smallest d s.t. ∃ d-partition sequence of G.

[Bonnet, K, Thomassé, Watrigant '20]

FO model checking can be done in time $f(d,|\phi|)\cdot n$

when a d-contraction sequence is given.

[Bonnet, K, Thomassé, Watrigant '20]

Input: a graph G, first-order sentence ϕ.
Question: $G \models \phi$?

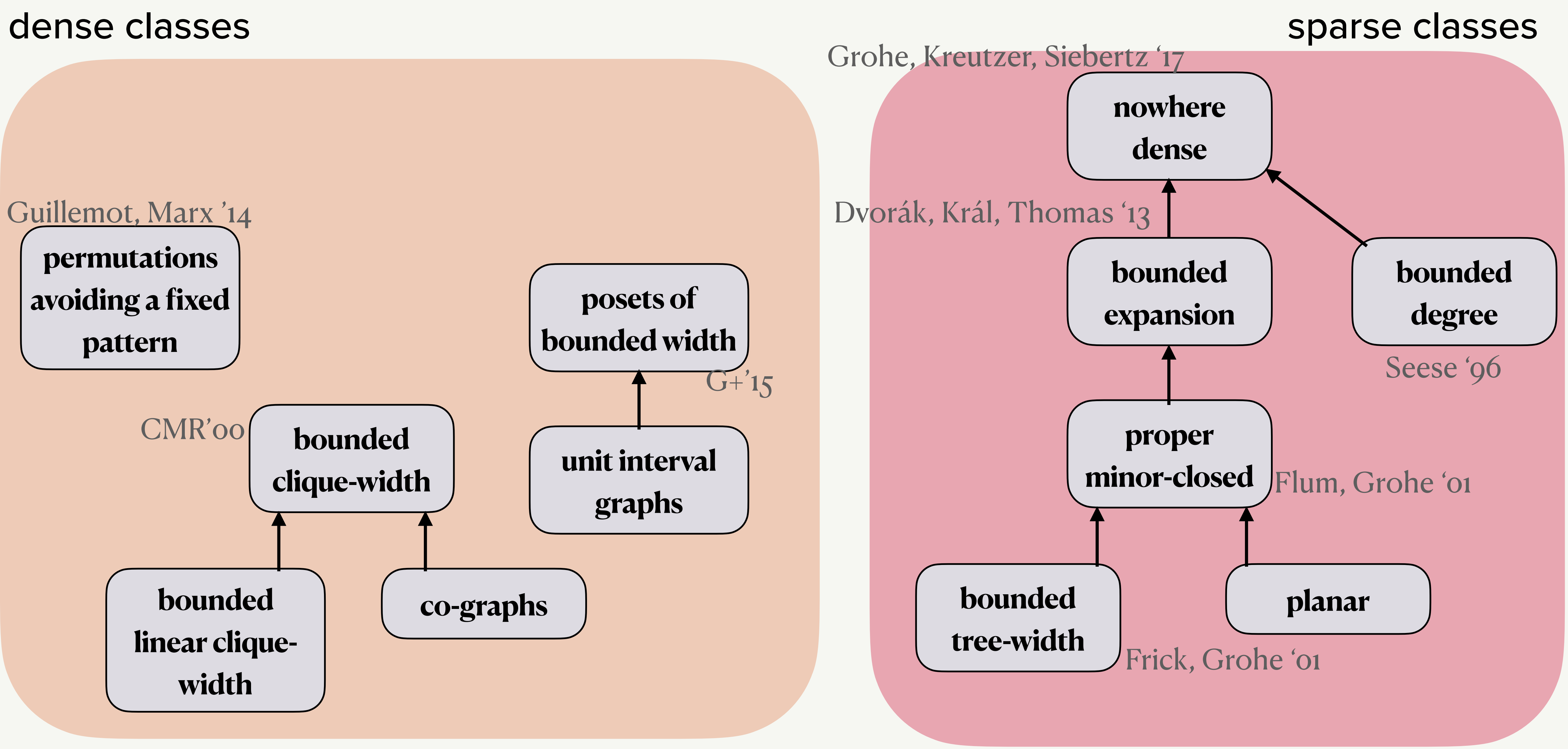
FO model checking can be done in time $f(d,|\phi|)\cdot n$

when a d-contraction sequence is given.

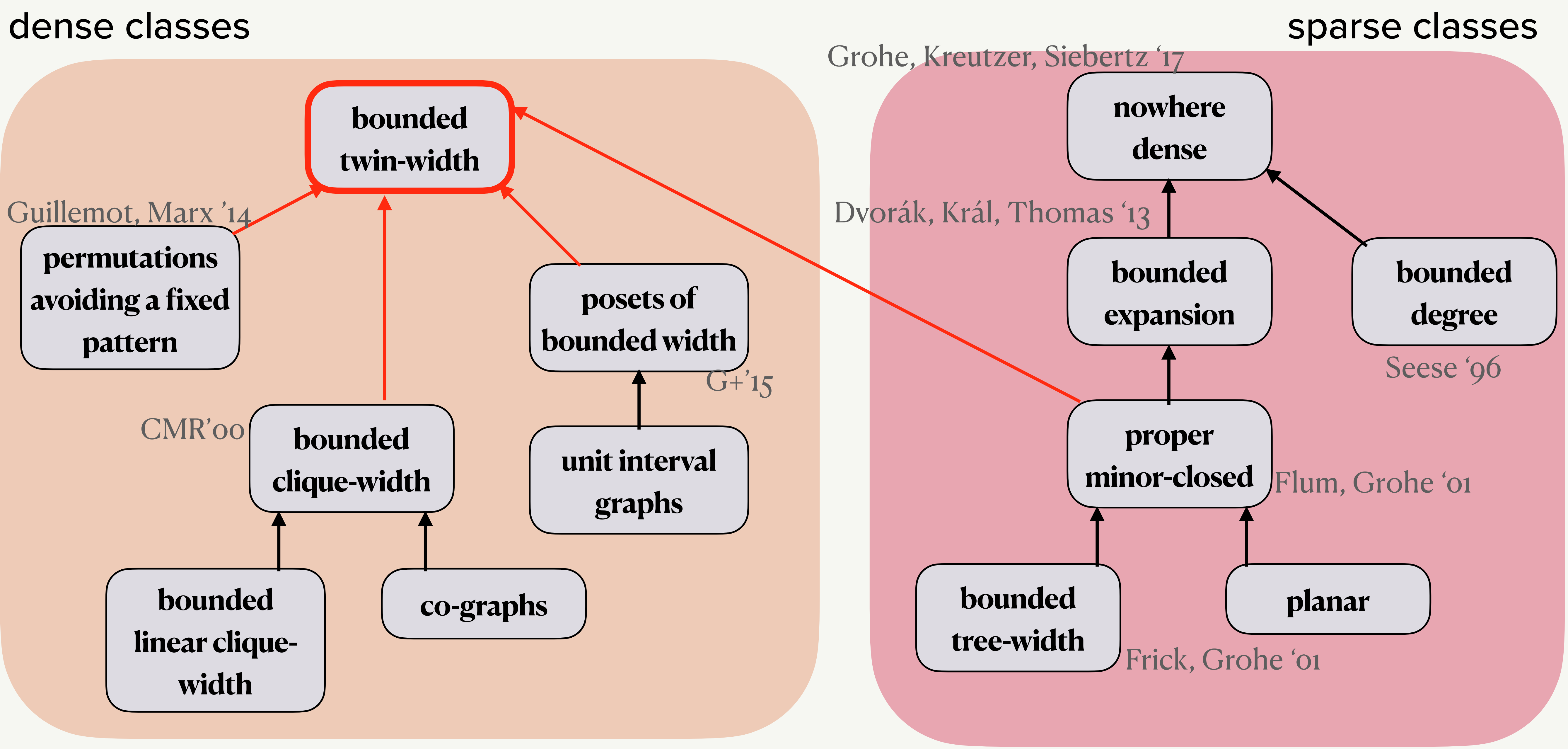
$$\Phi := \exists x_1 \exists x_2 \cdots \exists x_k \forall u \bigvee_{1 \leq i \leq k} \big((x_i = u) \vee E(x_i, u)\big)$$

$\rightsquigarrow G \models \Phi$ iff G has a dominating set of size k.

FO-model checking is FPT [BKTW'20]



FO-model checking is FPT [BKTW'20]



FO model checking algorithm when a d-partition sequence is given

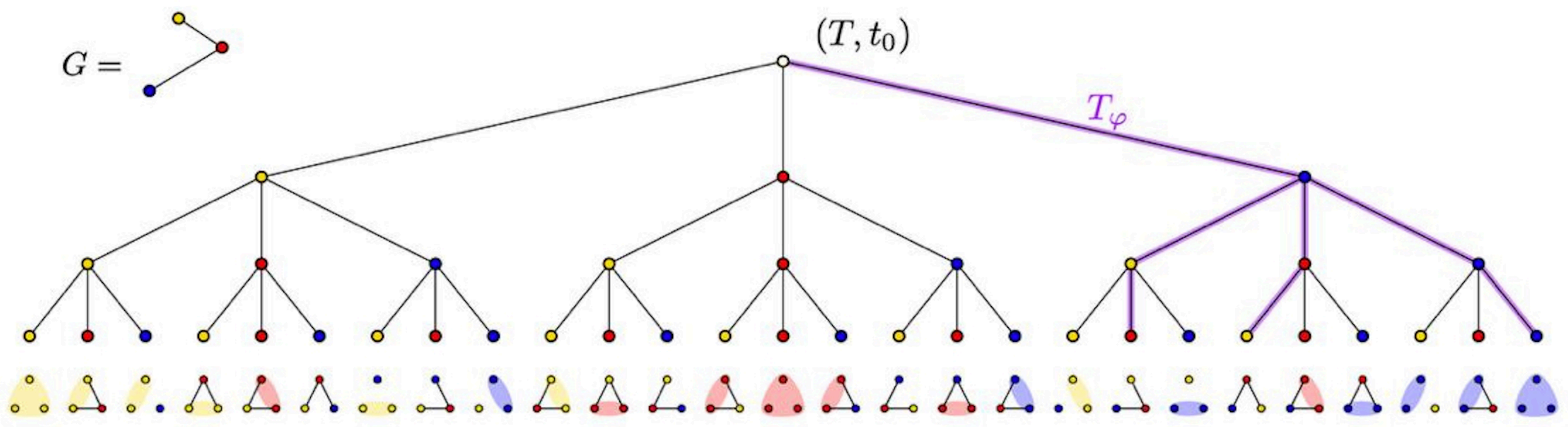
Prenex Normal Form

$$\varphi = Q_1 x_1 Q_2 x_2 \cdots Q_\ell x_\ell \phi^*$$

- each Q_i is a non-negated quantifier (∀, ∃)
- ϕ^* is a quantifier-free sentence; a boolean combination of $(x_i = x_j)$ and $E(x_i, x_j)$
- Any FO-sentence of quantifier rank q can be rewritten as a prenex sentence of depth $f(q)$ for some f.
- We assume that the FO sentence we want to test is given in prenex form.

ℓ-Morphism Tree (Game tree) in G

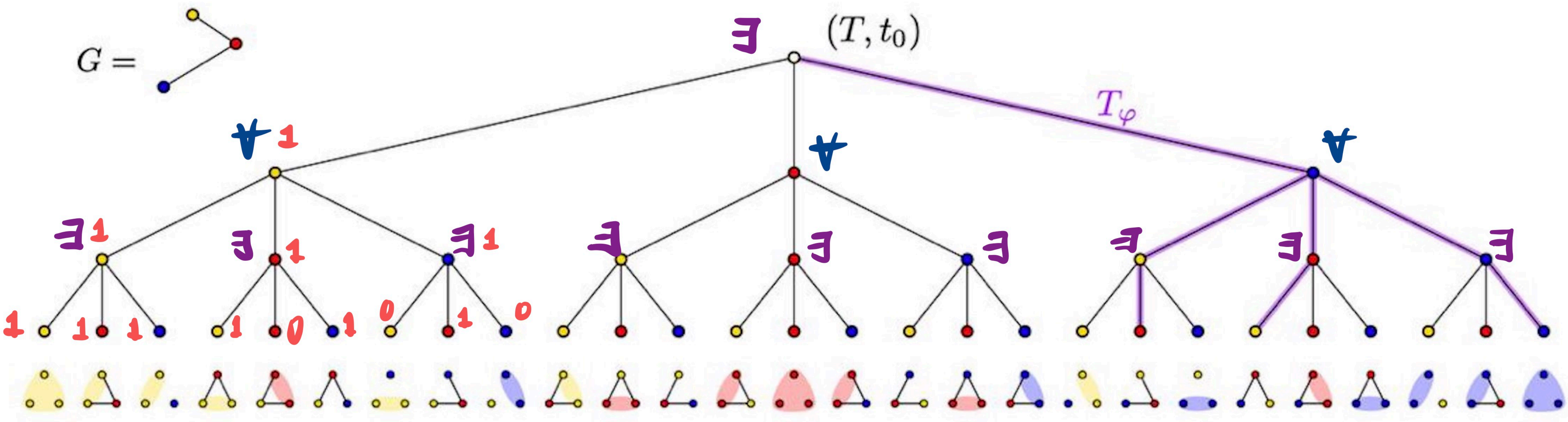
$$\varphi = \exists x_1 \forall x_2 \exists x_3 (x_1 = x_2 \vee E(x_2, x_3))$$



- all possible ℓ-tuples of vertices can be described as a game tree rooted at ε, called a complete ℓ-morphism tree $MT_\ell(G)$.
- For any prenex sentence φ of depth ℓ, $G \models \varphi$ can be tested using $MT_\ell(G)$.

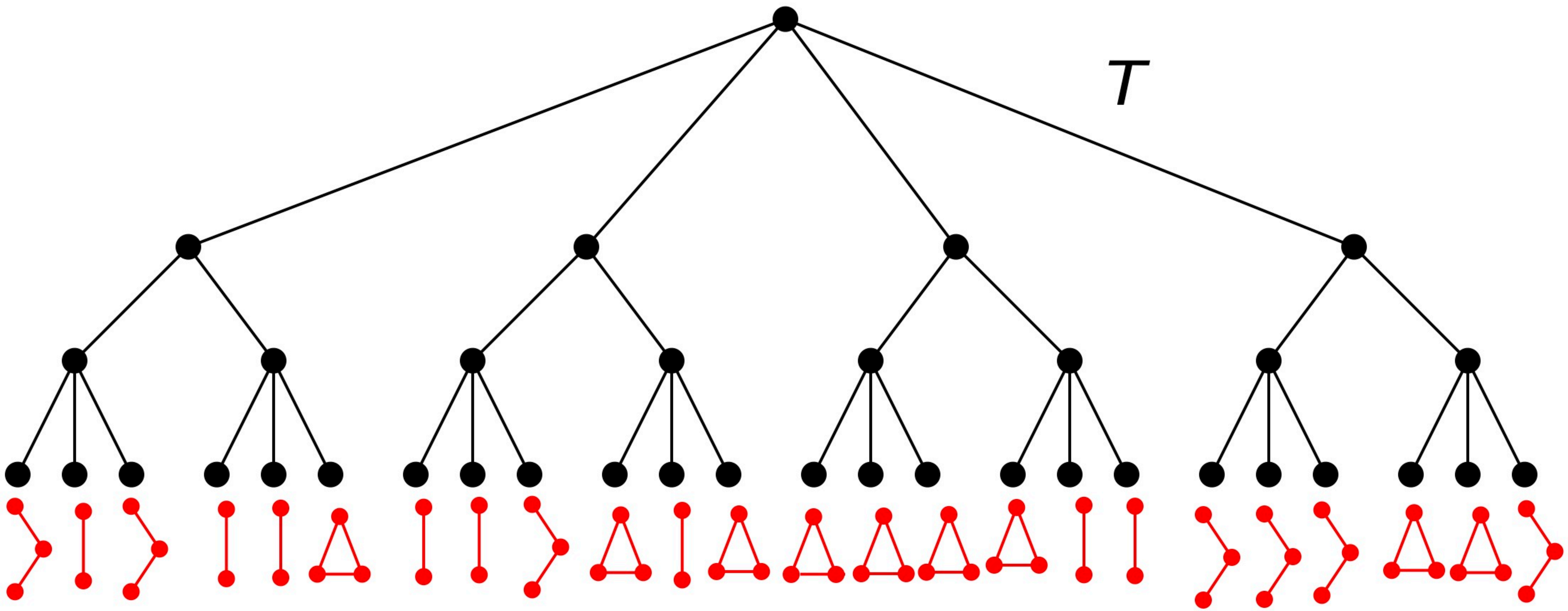
Testing $G \models \varphi$ using $MT_\ell(G)$

$$\varphi = \exists x_1 \forall x_2 \exists x_3 (x_1 = x_2 \vee E(x_2, x_3))$$

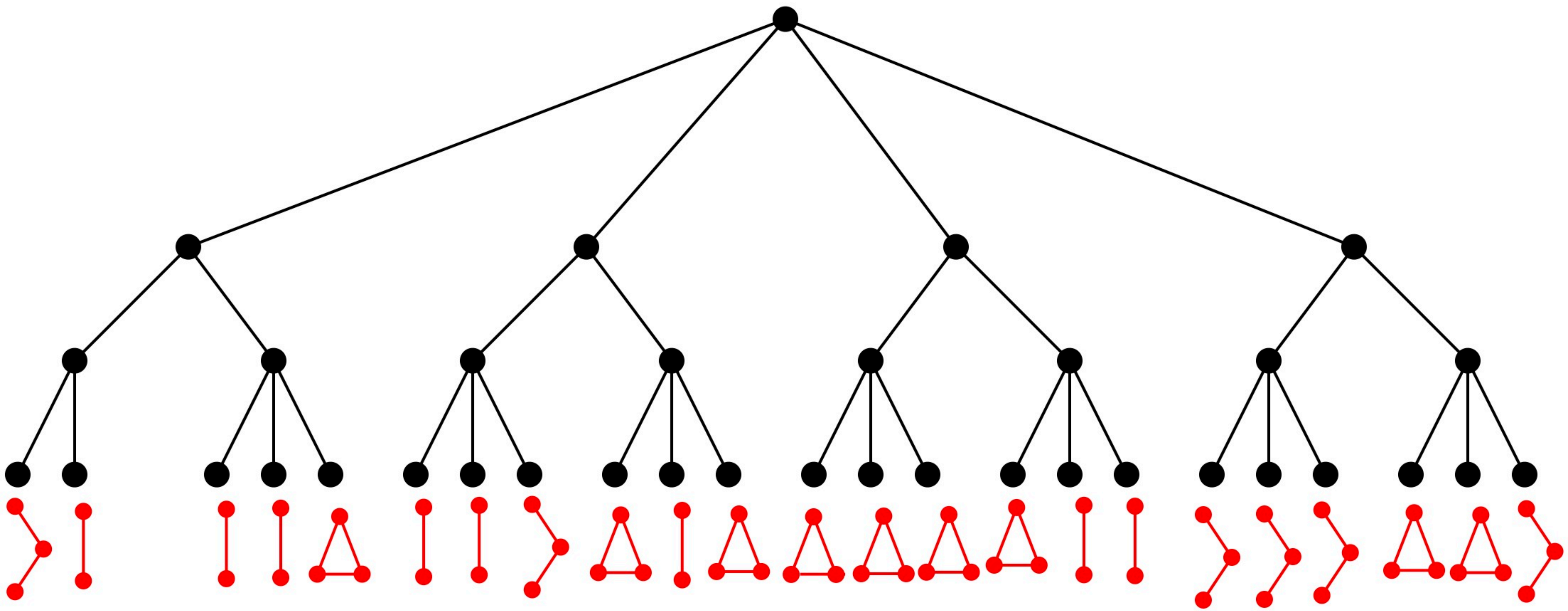


- $MT_\ell(G)$ has size n^ℓ. Let's reduce the size to make it more useful.

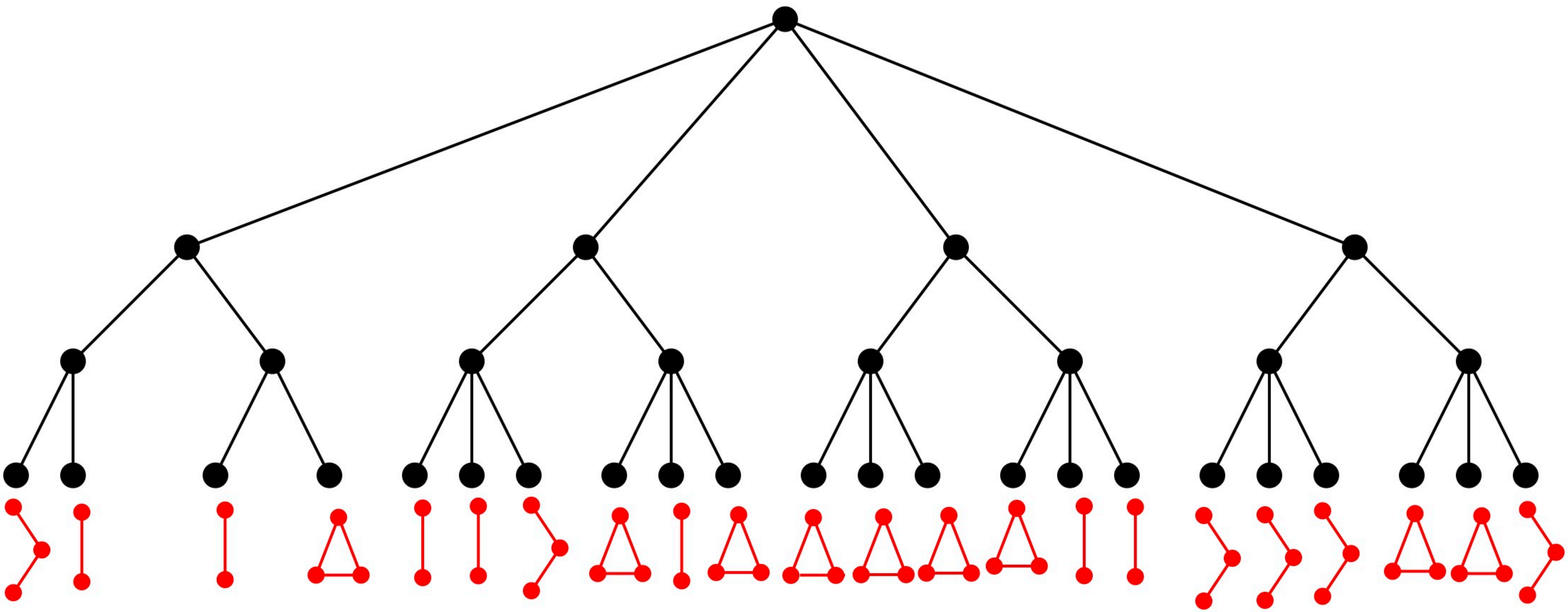
(Full) Reduction of ℓ-Morphism Tree



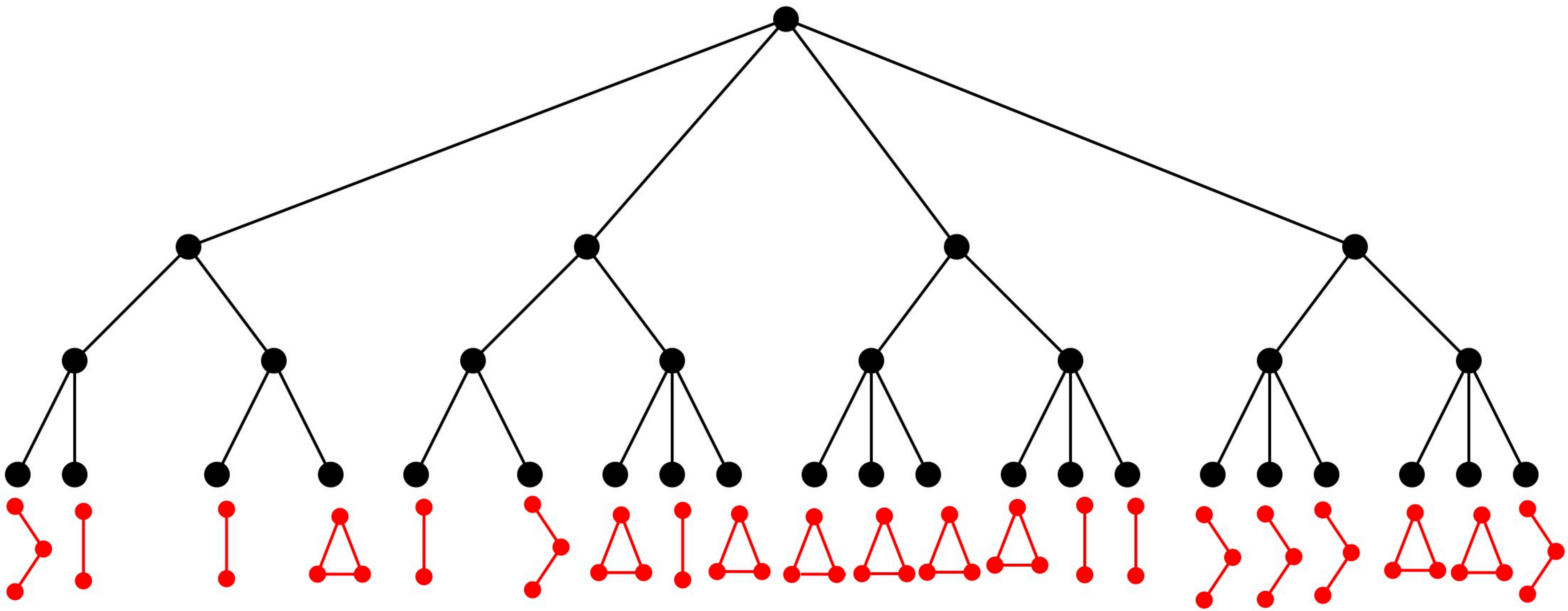
(Full) Reduction of ℓ-Morphism Tree



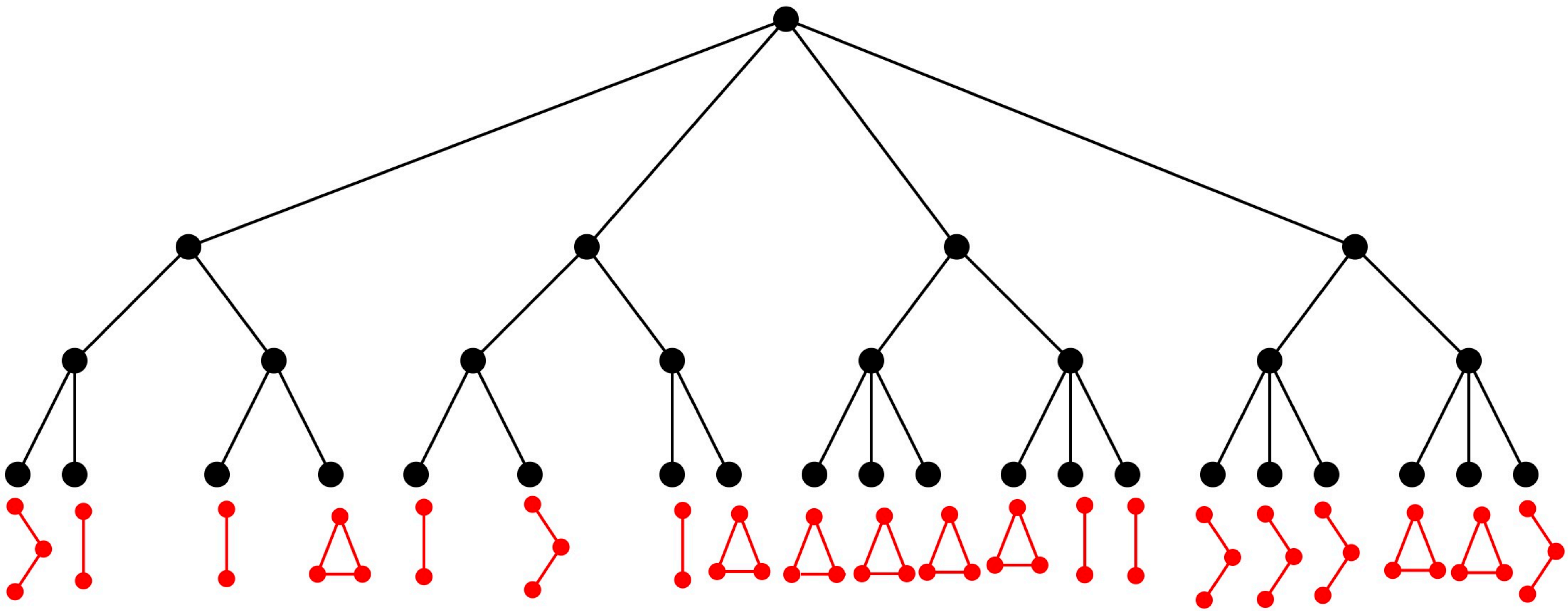
(Full) Reduction of ℓ-Morphism Tree



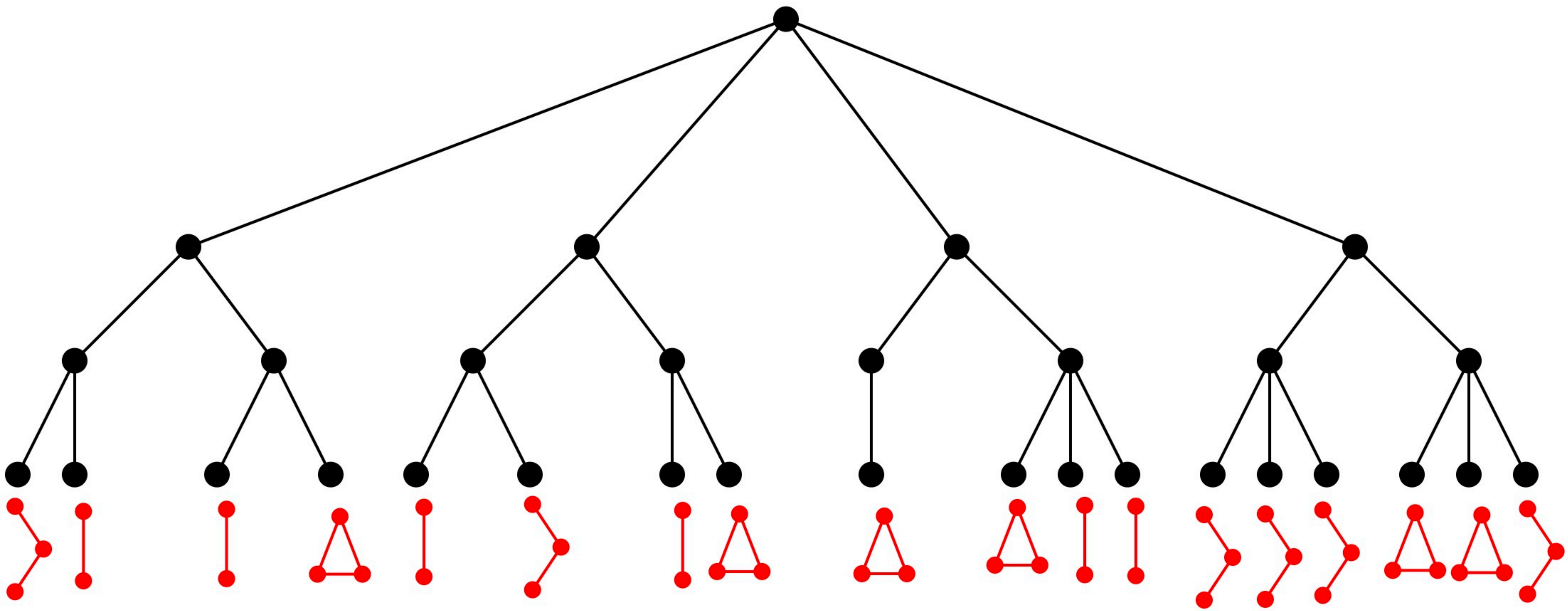
(Full) Reduction of ℓ-Morphism Tree



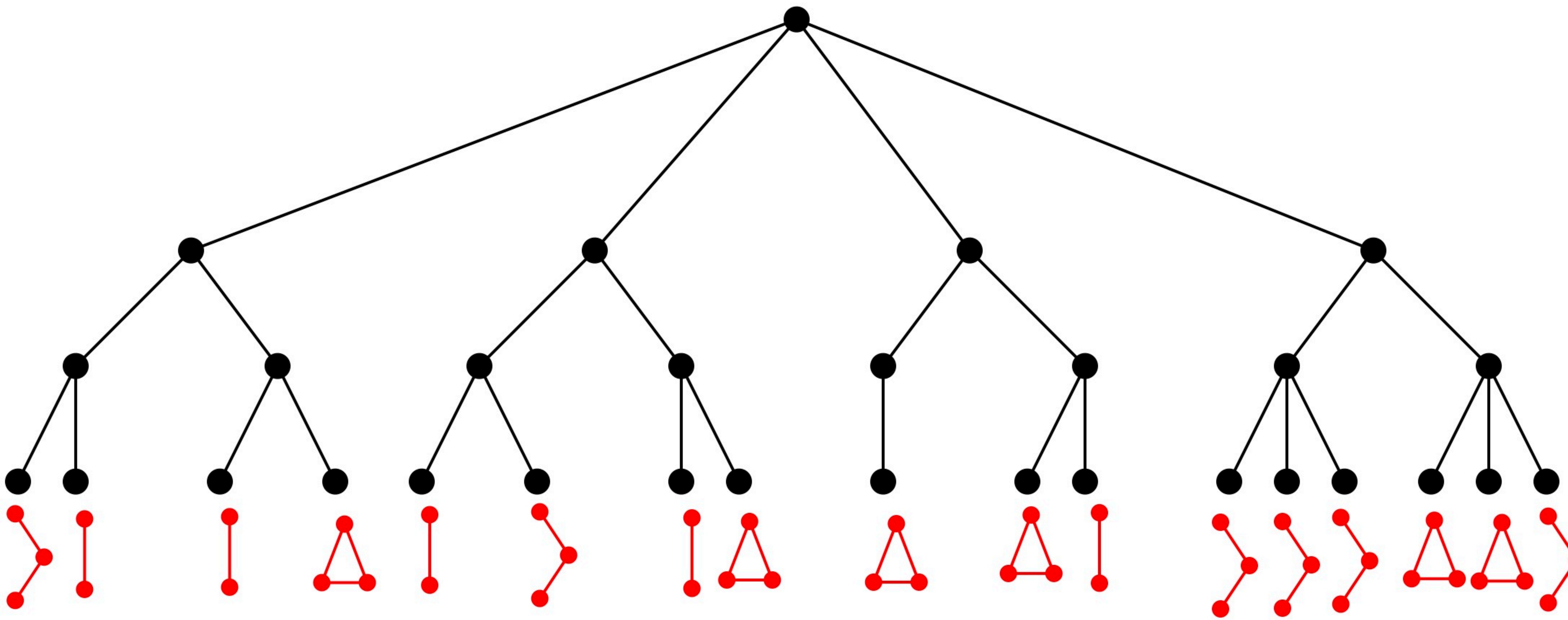
(Full) Reduction of ℓ-Morphism Tree



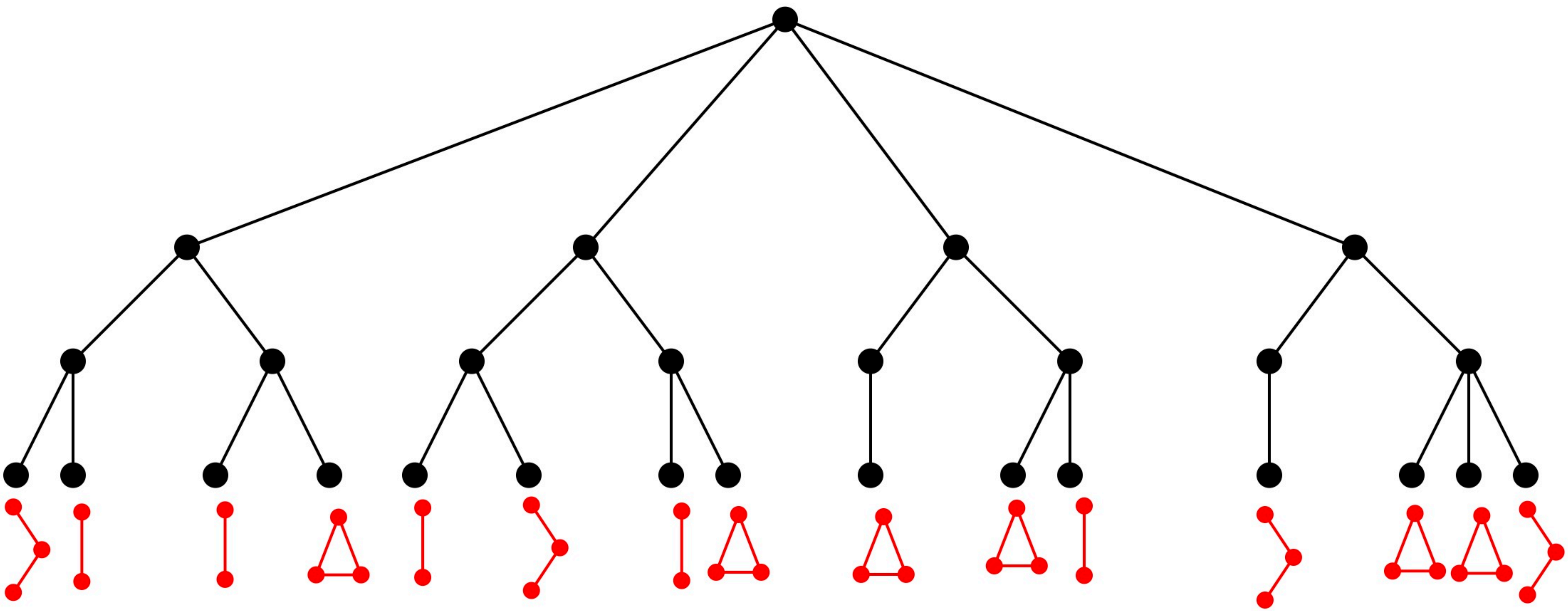
(Full) Reduction of ℓ-Morphism Tree



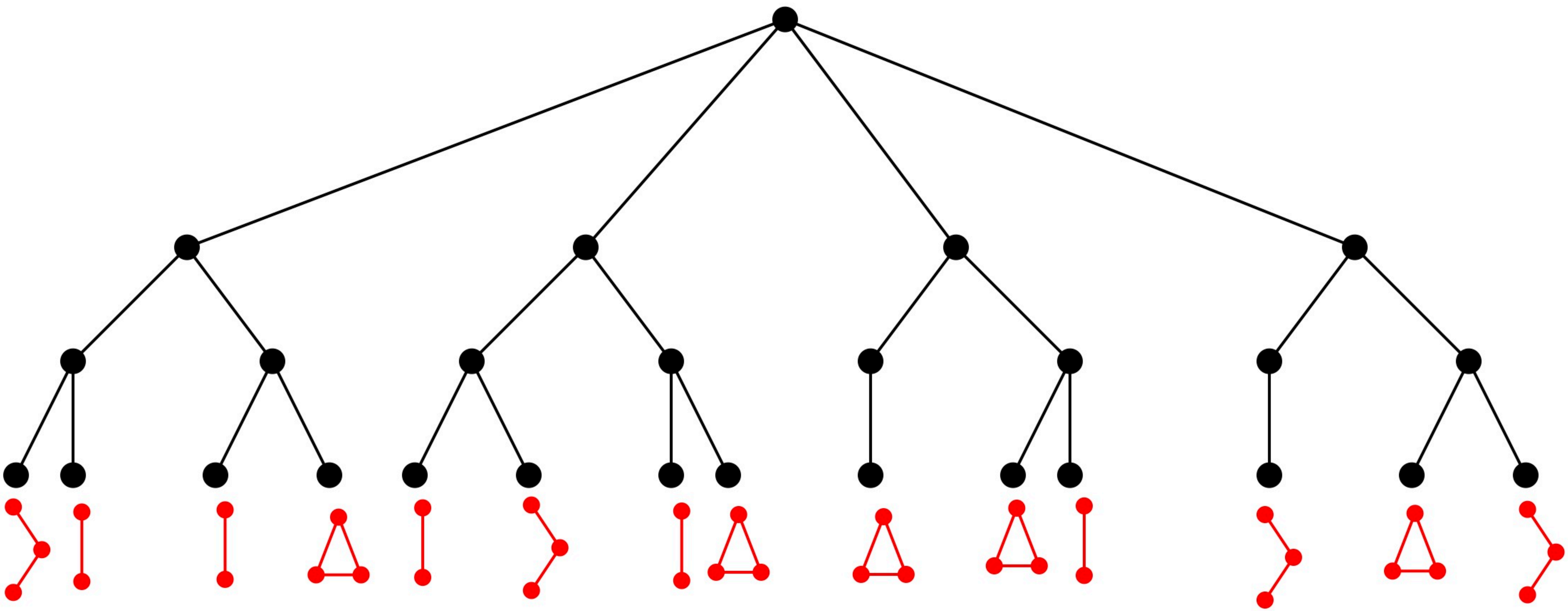
(Full) Reduction of ℓ-Morphism Tree



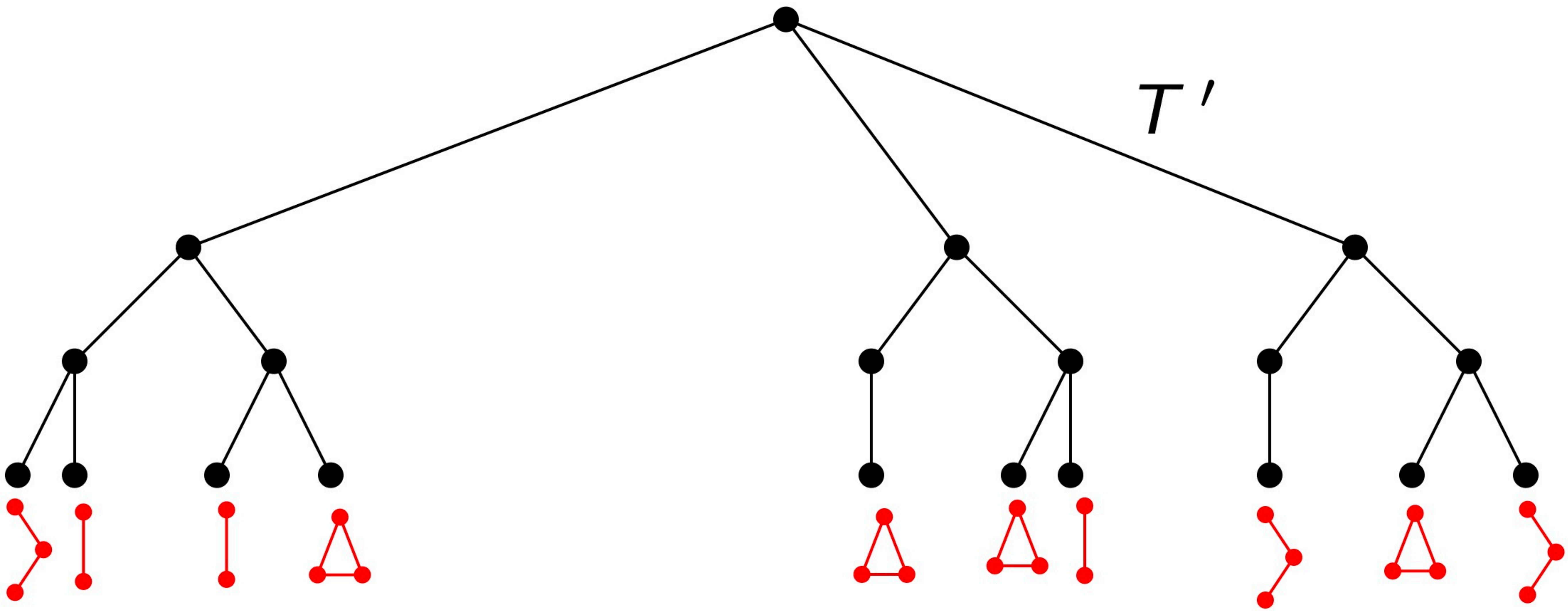
(Full) Reduction of ℓ-Morphism Tree



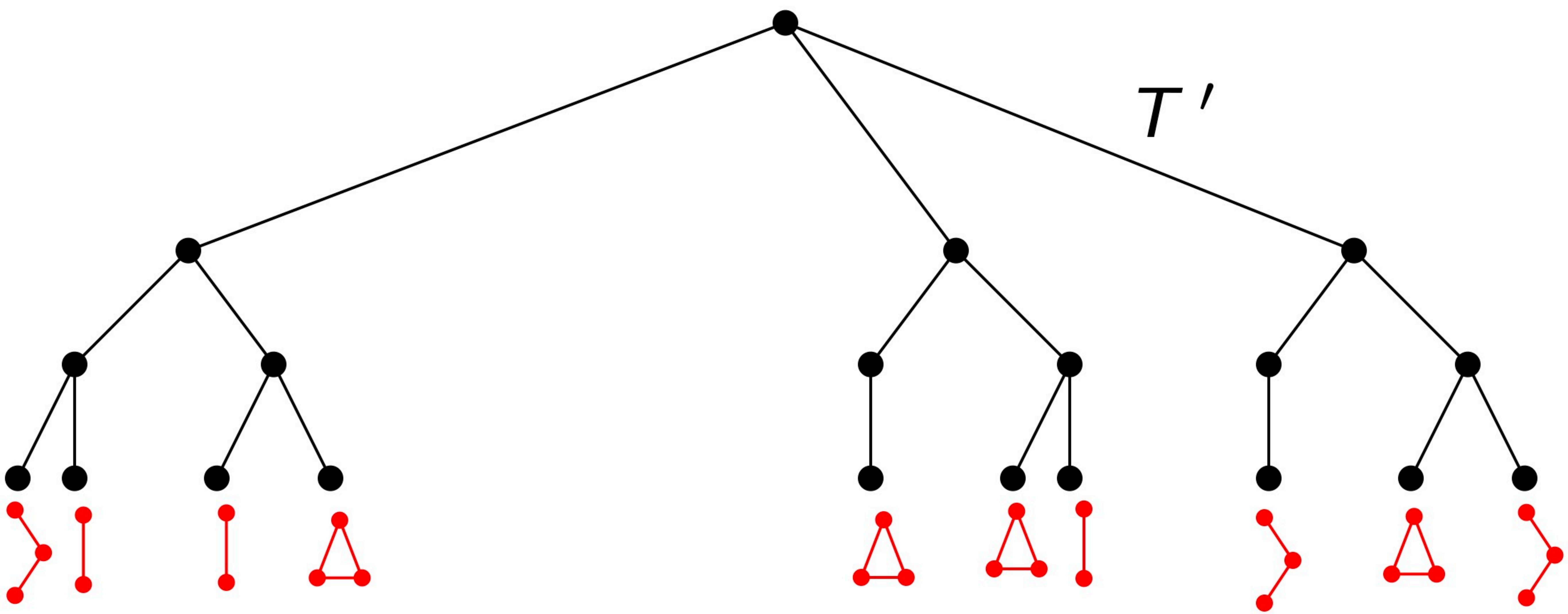
(Full) Reduction of ℓ-Morphism Tree



(Full) Reduction of ℓ-Morphism Tree



(Full) Reduction of ℓ-Morphism Tree



Full Reduction $MT'_\ell(G)$ of ℓ-Morphism Tree $MT_\ell(G)$

- The size of a full reduction $MT'_\ell(G)$ is bounded by a function of ℓ.
- If $MT'_\ell(G_1) = MT'_\ell(G_2)$ ➔ G_1 and G_2 satisfies precisely the same set of prenex FO sentences of depth $\leq \ell$.

- In general, we cannot compute $MT'_\ell(G)$ efficiently.
- We show that $MT'_\ell(G)$ can be computed in time $f(d, \ell) \cdot n$ when a d-partition sequence is given.

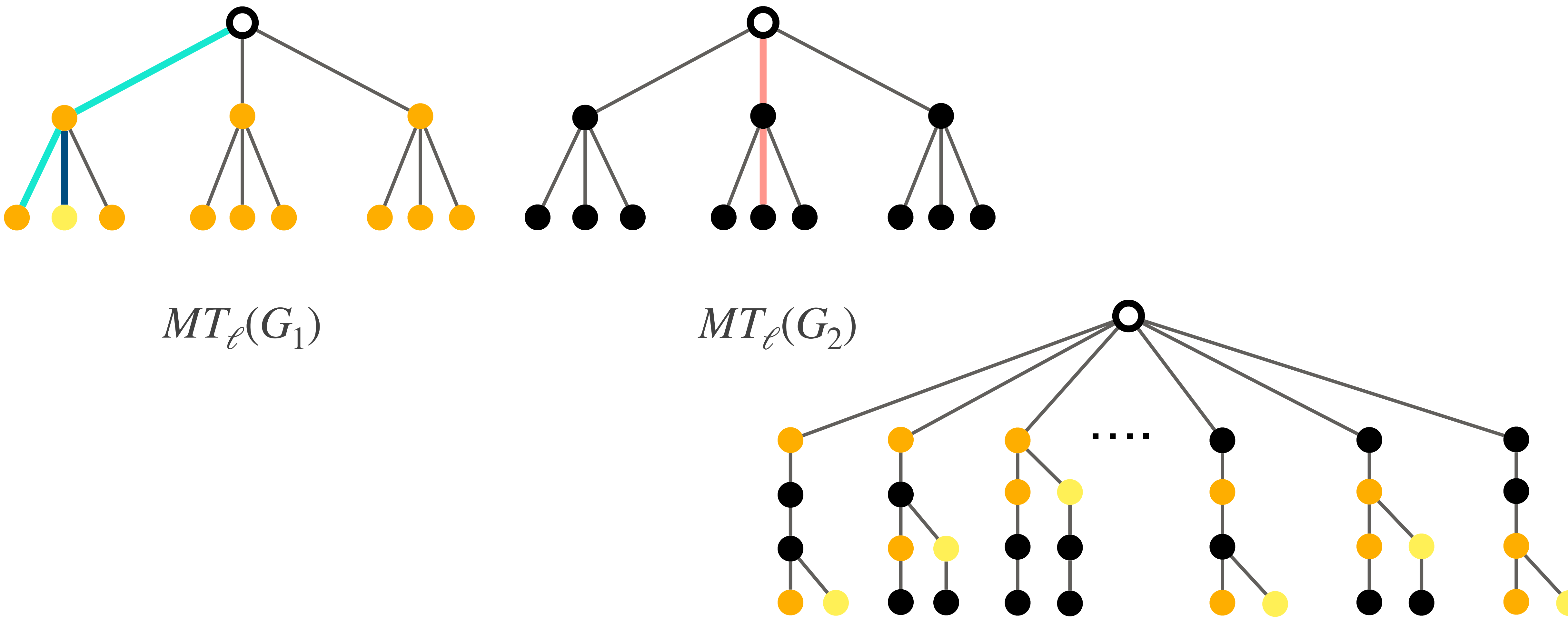
Strategy: first attempt

Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$

- Following the d-partition sequence $\mathscr{P}_n, \cdots, \mathscr{P}_1$
- At $\mathscr{P}_i$: maintain the list of $MT'_\ell(G[X])$ for each $X \in \mathscr{P}_i$
- At $\mathscr{P}_1 = \{V\}$: $MT'_\ell(G[V]) = MT'_\ell(G)$

Strategy: first attempt

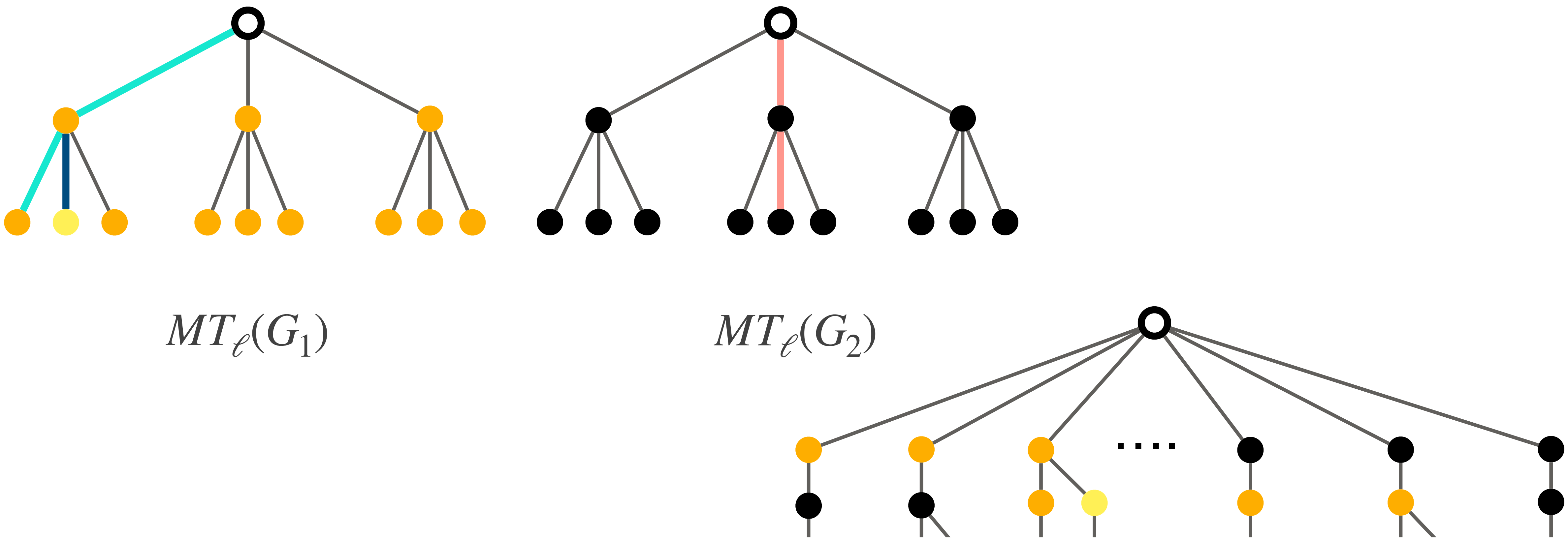
Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$



$MT_\ell(G)$ can be obtained by "shuffling" all pairs of root-to-leaf paths and arranging them by prefix relations, then truncate all nodes of depth $> \ell$.

Strategy: first attempt

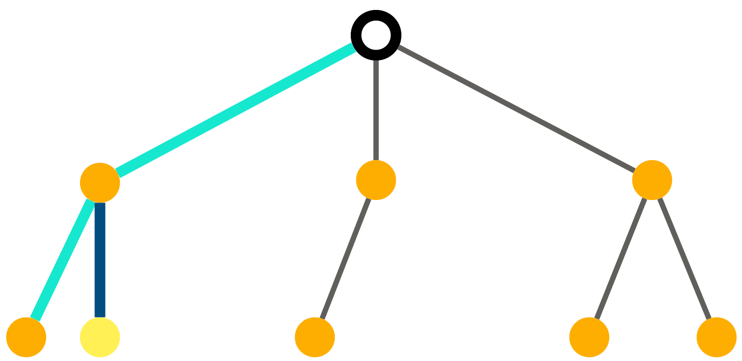
Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$



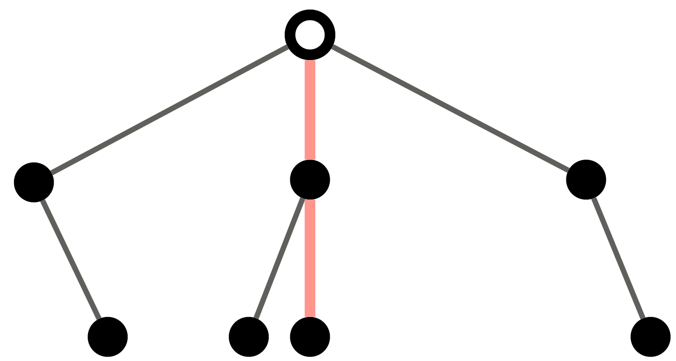
$MT_\ell(G)$ can be obtained by "shuffling" all pairs of root-to-leaf paths and arranging them by prefix relations, then truncate all nodes of depth $> \ell$.

Strategy: first attempt

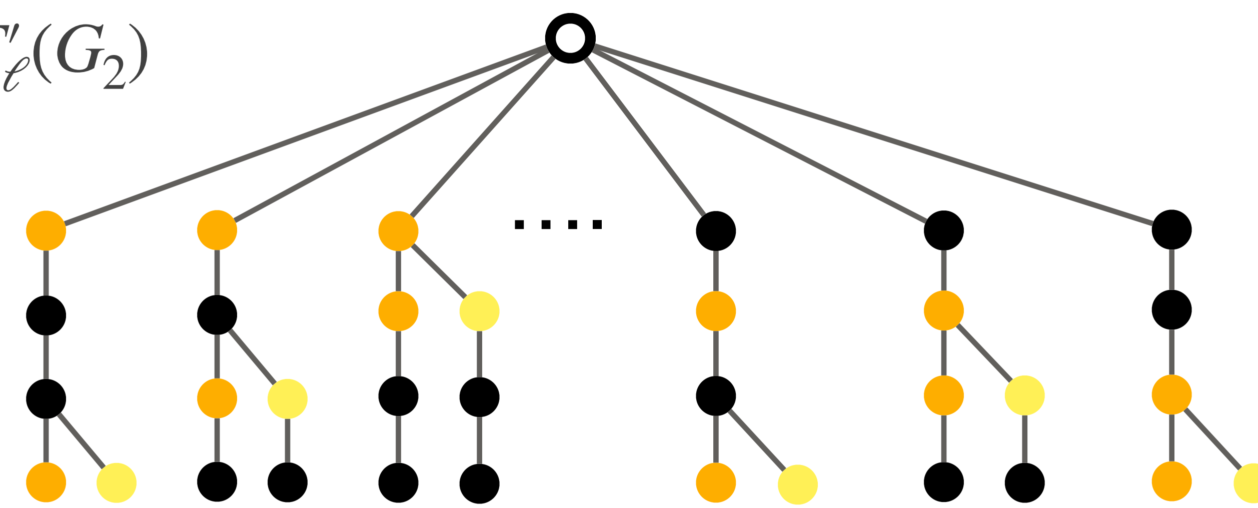
Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$



$MT'_\ell(G_1)$



$MT'_\ell(G_2)$



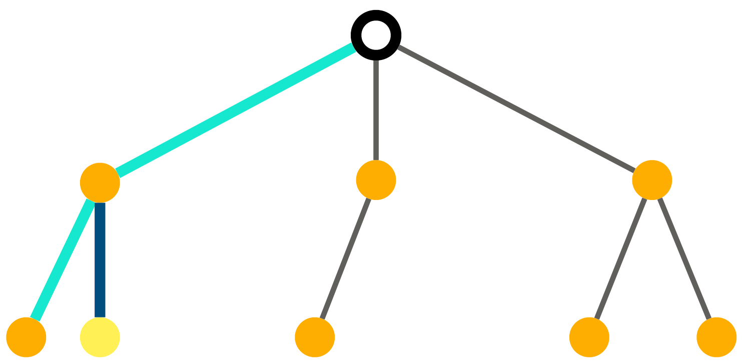
With ℓ-shuffle of (fully reduced) $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$, do we not lose information? That is,

ℓ-shuffle of $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$ is a reduction of $MT_\ell(G)$?

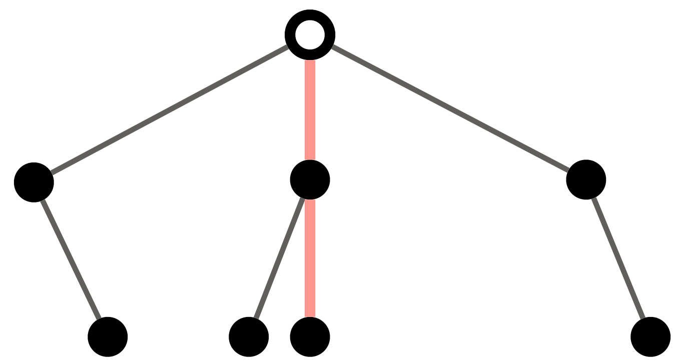
Yes, if it is fully (non-)adjacent between $V(G_1)$ and $V(G_2)$.

Strategy: first attempt

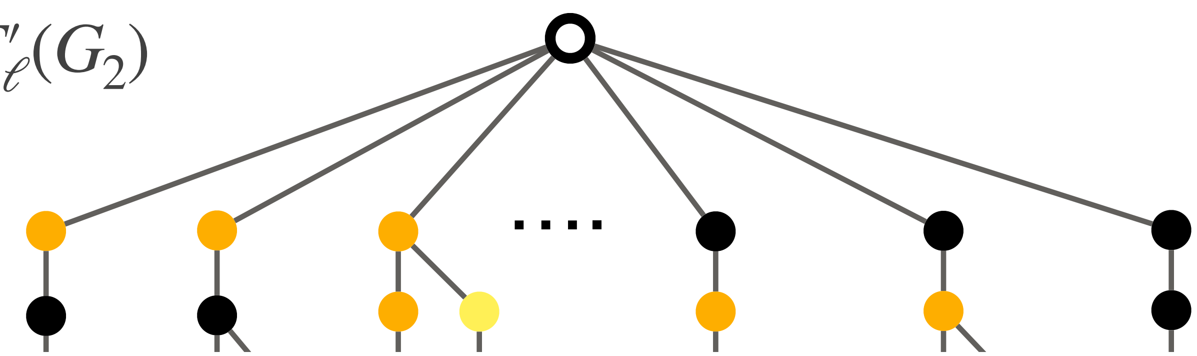
Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$



$MT'_\ell(G_1)$



$MT'_\ell(G_2)$



With ℓ-shuffle of (fully reduced) $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$,
do we not lose information? That is,
ℓ-shuffle of $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$ is a reduction of $MT_\ell(G)$?
Yes, if it is fully (non-)adjacent between $V(G_1)$ and $V(G_2)$.

Strategy: first attempt

Maintain $MT'_\ell(G[X])$ per part $X \in \mathscr{P}$

"ℓ-shuffle" $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$ and reduce the obtained ℓ-morphism tree.

➔ By doing this, we do not lose any information for $MT_\ell(G_1 \oplus G_2)$ (or $MT_\ell(G_1 \otimes G_2)$).

➔ i.e. ℓ-shuffle of $MT'_\ell(G_1)$ and $MT'_\ell(G_2)$ is a reduction of the full morphism-tree $MT_\ell(G)$

➔ This works for 0-partition sequence (i.e. cographs, **but not with d-partition sequence in general.**

Strategy

Maintain $MT'_\ell(G, \mathscr{P}, X)$ per part $X \in \mathscr{P}$

$MT_\ell(G, \mathscr{P}, X)$ concerns only the game move $(a_1, \ldots, a_\ell)$ in $(X_1, \ldots, X_\ell) \in \mathscr{P}^\ell$ s.t.

- $X_1 = X$
- $\mathrm{dist}_{G_\mathscr{P}}(X, X_i) \leq 3^\ell$ (minus some technical condition to guarantee an efficient update of the $MT'_\ell(G, \mathscr{P}, X)$'s after contraction)

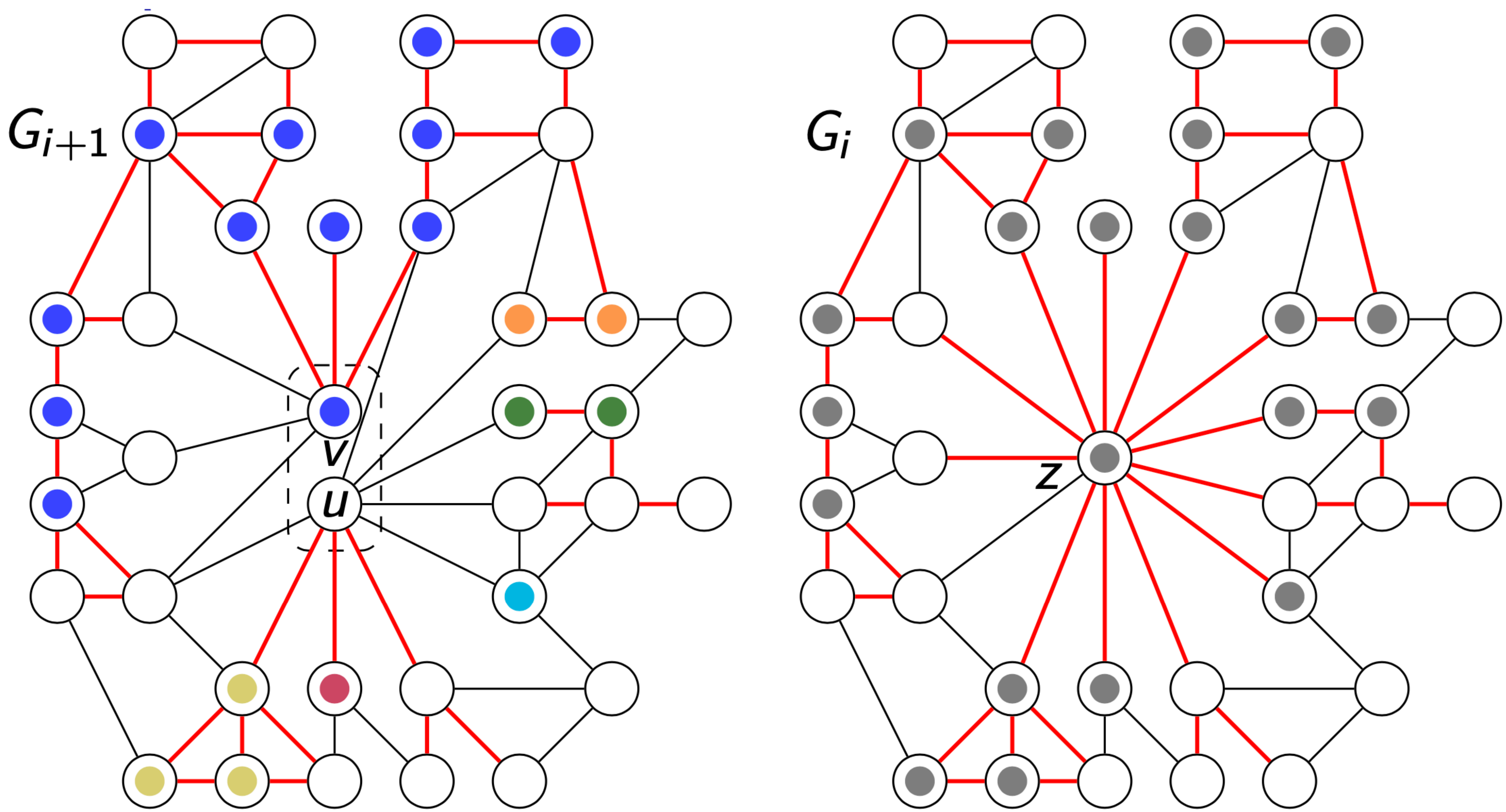
To reduce to $MT'_\ell(G, \mathscr{P}, X)$, the isomorphism between two siblings take into account the membership in parts of $\mathscr{P}$.

The size of $MT'_\ell(G, \mathscr{P}, X)$ is bounded by some function $h(\ell, d)$ as the number of distinct parts in the radius 3^ℓ-ball centered at X is bounded ($\leq d^{3^\ell} + 1$).

Strategy: update from $\mathscr{P}_{i+1}$ to $\mathscr{P}_i$

Maintain $MT'_\ell(G, \mathscr{P}, X)$ per part $X \in \mathscr{P}$

$X_u, X_v \in \mathscr{P}_{i+1}$ are merged to form a part X_z, to yield $\mathscr{P}_i$



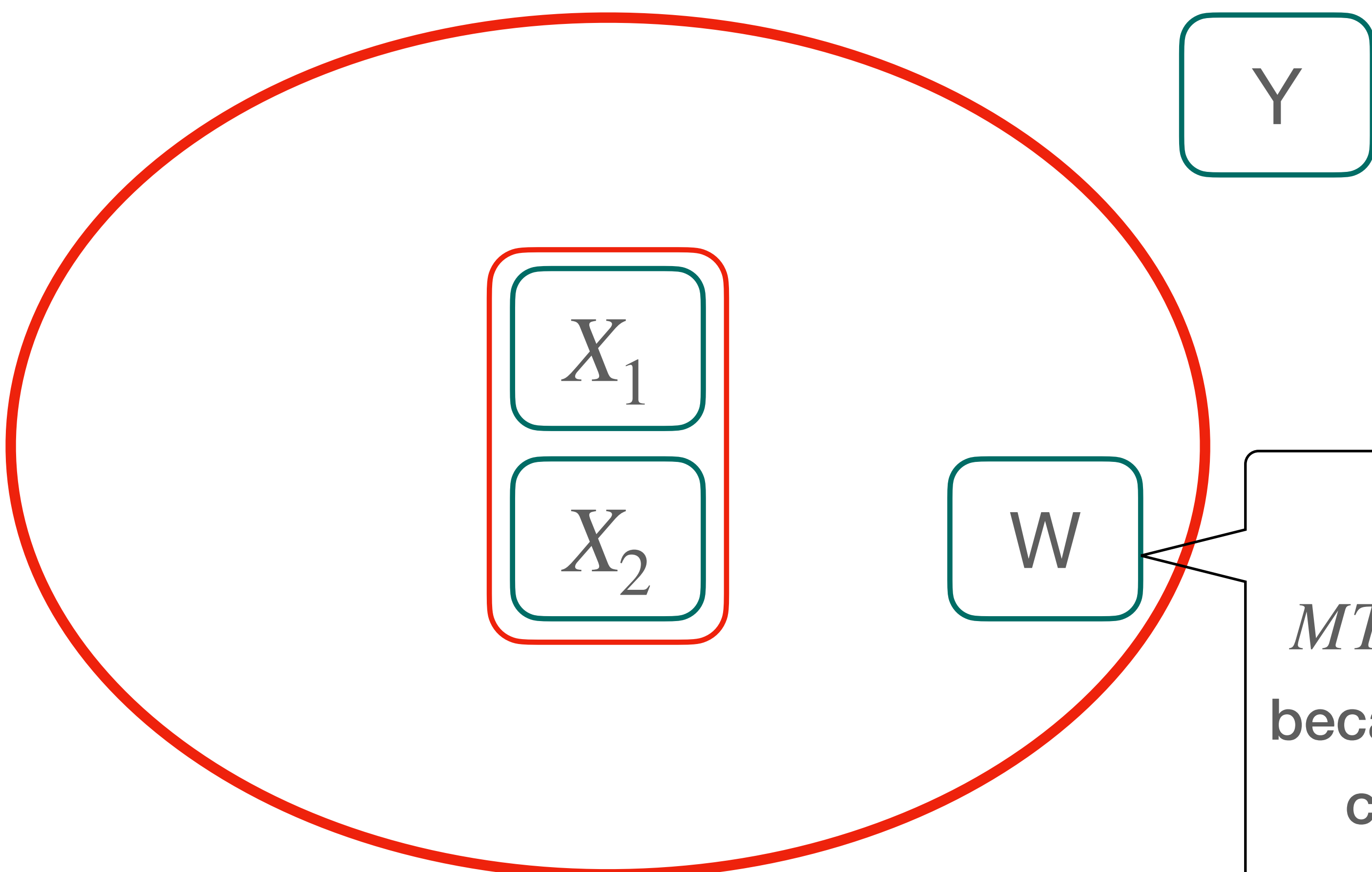
$G_{\mathscr{P}_{i+1}}$ induced by the ball R

radius-3^ℓ ball R
centered at $X_1 \cup X_2$ in the red graph $G_{\mathscr{P}_i}$

Strategy: update from $\mathscr{P}_{i+1}$ to $\mathscr{P}_i$

Maintain $MT'_\ell(G, \mathscr{P}, X)$ per part $X \in \mathscr{P}$

radius-3^ℓ ball R centered at $X_1 \cup X_2$ in the red graph $G_{\mathscr{P}_i}$



$MT'_\ell(G, \mathscr{P}_i, Y) = MT'_\ell(G, \mathscr{P}_{i+1}, Y)$
for $Y \in \mathscr{P}_i$
which is 3^ℓ-away from $X_1 \cup X_2$ in $G_{\mathscr{P}_i}$

for the parts W in the ball R,
$MT'_\ell(G, \mathscr{P}, W)$ needs to be recomputed
because 3^ℓ-ball centered at W might have
changed in $G_{\mathscr{P}_i}$ even if $W \neq X_1 \cup X_2$

To compute $MT'_\ell(G, \mathscr{P}_i, W)$, we ℓ-shuffle over the parts Z in R
'sufficiently far' from W in $G_{\mathscr{P}_{i+1}}$:
the information from $MT'_\ell(G, \mathscr{P}_{i+1}, Z)$ for Z close to W are already implemented
in $MT'_\ell(G, \mathscr{P}_{i+1}, W)$.

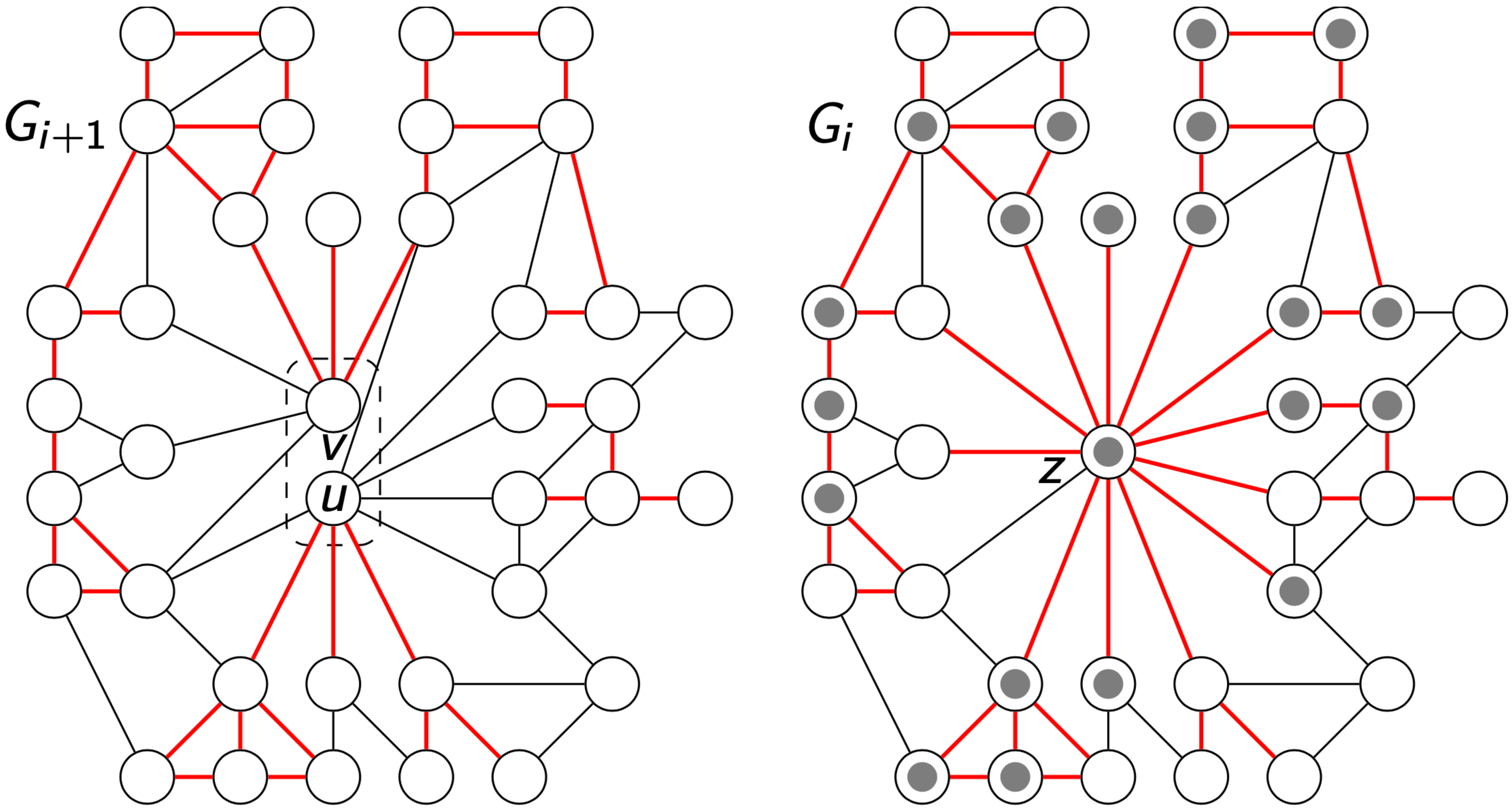
Recap of FO model checking algorithm

For φ in prenex form of depth ℓ; almost true version

- Follow the d-partition sequence $\mathscr{P}_n, \mathscr{P}_{n-1}, \ldots, \mathscr{P}_1$.
- Initialization for $\mathscr{P}_n$: $G_{\mathscr{P}_n}$ is edgeless. The fully reduced ℓ-morphism tree $MT'_\ell(G, \mathscr{P}_n, \{v\})$ is a length-ℓ path, each node corresponding to $(v), (v, v), \cdots$ and $(v, \ldots, v)$
- Assume for $\mathscr{P}_{i+1}$: $MT'_\ell(G, \mathscr{P}_{i+1}, X)$ is given for each $X \in \mathscr{P}_{i+1}$
- $\mathscr{P}_i = \mathscr{P}_{i+1} \backslash \{X_1, X_2\} \cup \{X_1 \cup X_2\}$:
  - $R = N^{3^\ell}_{G_{\mathscr{P}_i}}(X_1 \cup X_2)$
  - If $Y \notin R$: $MT'_\ell(G, \mathscr{P}_i, X) := MT'_\ell(G, \mathscr{P}_{i+1}, X)$
  - If $Y \in R$: $MT'_\ell(G, \mathscr{P}_i, Z)$ is the ℓ-shuffle of all $MT'_\ell(G, \mathscr{P}_i, W)$ for $W \in R$ which was far from Z in $G_{\mathscr{P}_{i+1}}$.
- Check φ on $MT'_\ell(G, \mathscr{P}_1, V(G))$.

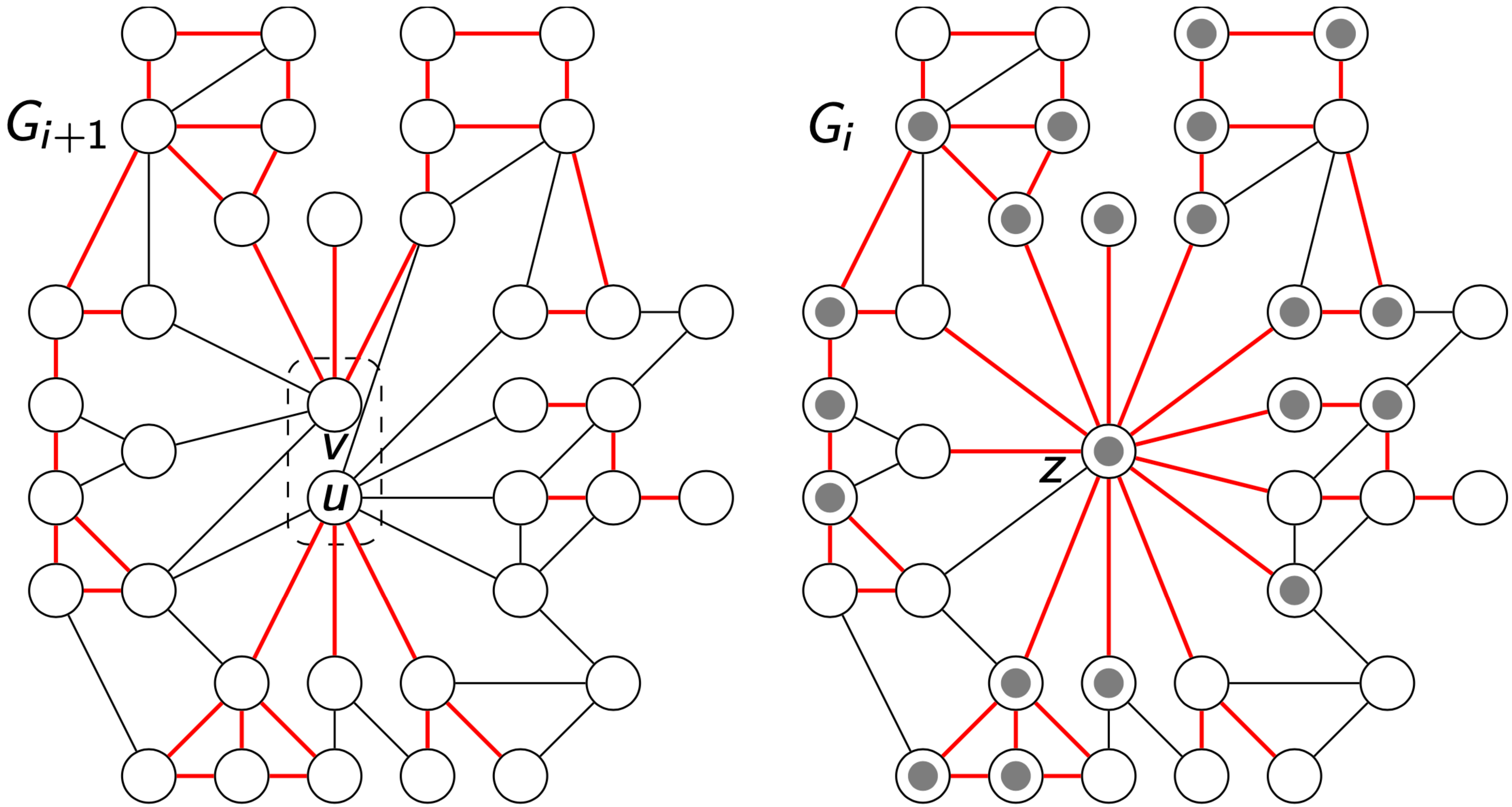
Example: k-Independent Set

T



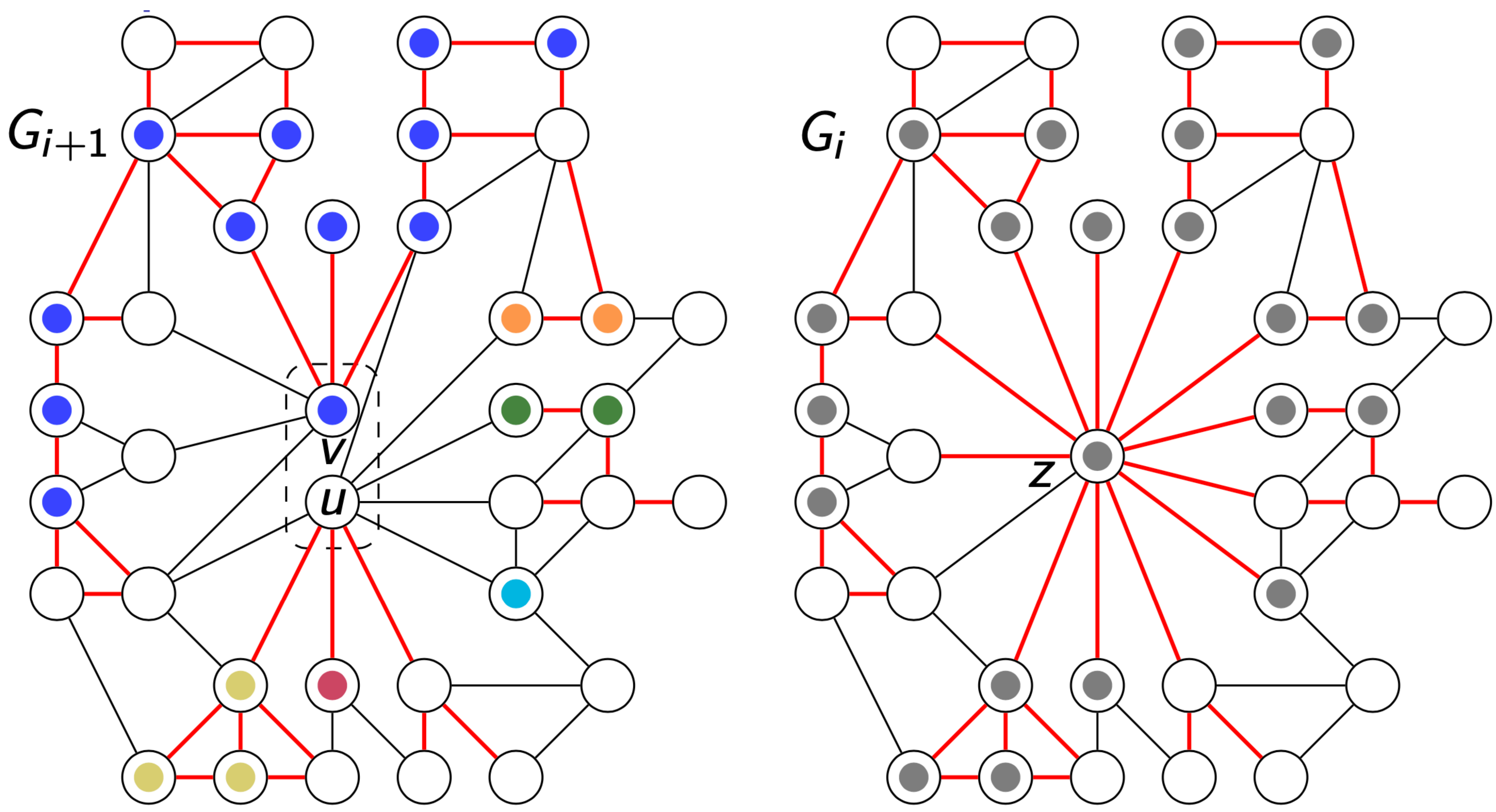
Example: k-Independent Set

T



For any partial solution S realizing T, three possibilities:

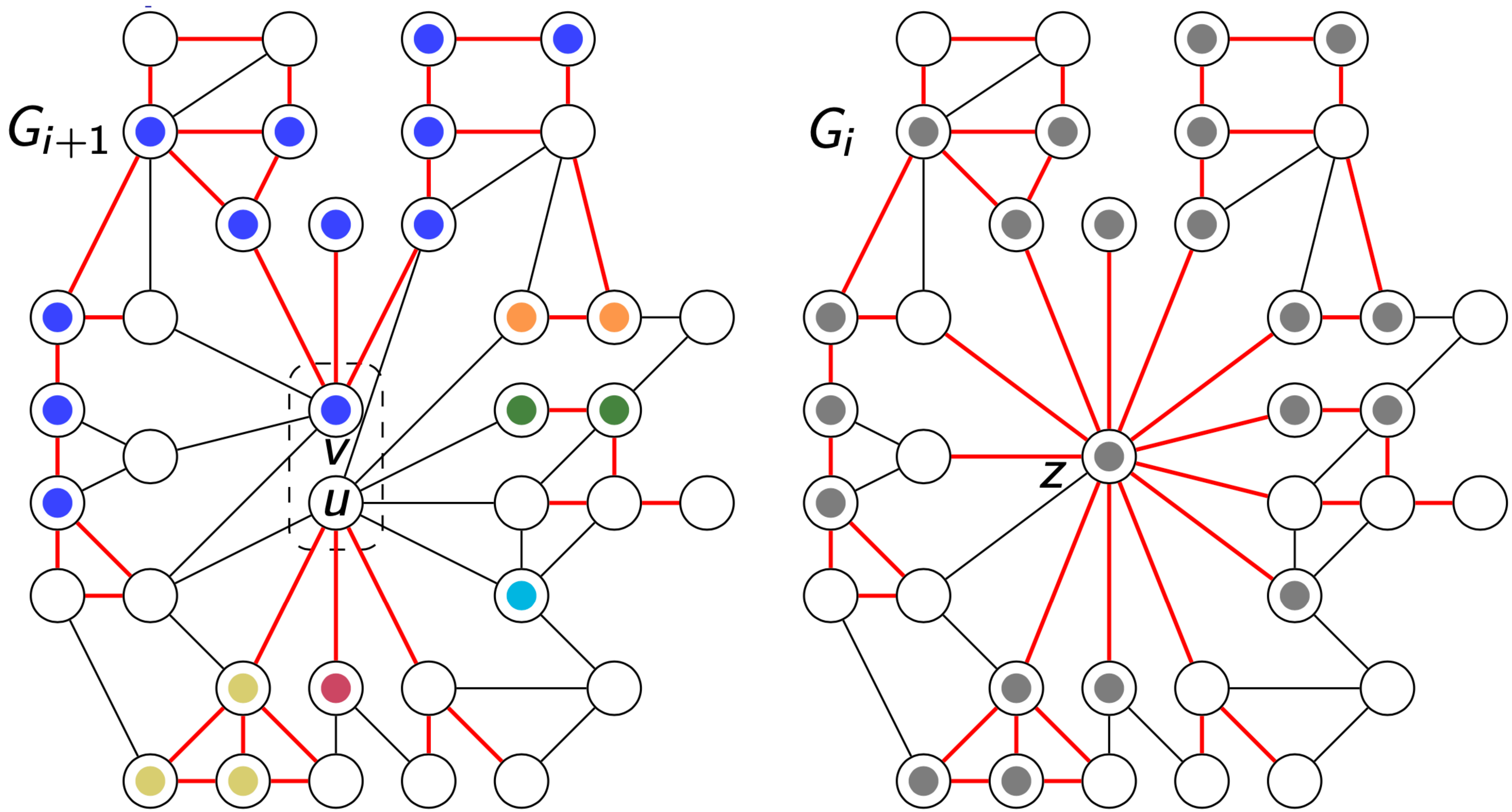
(a) $T \cap G_{i+1}(u) = \emptyset$, (b) $T \cap G_{i+1}(v) = \emptyset$, (c) both sets non-empty.



G_{i+1}
v
u
G_i
z

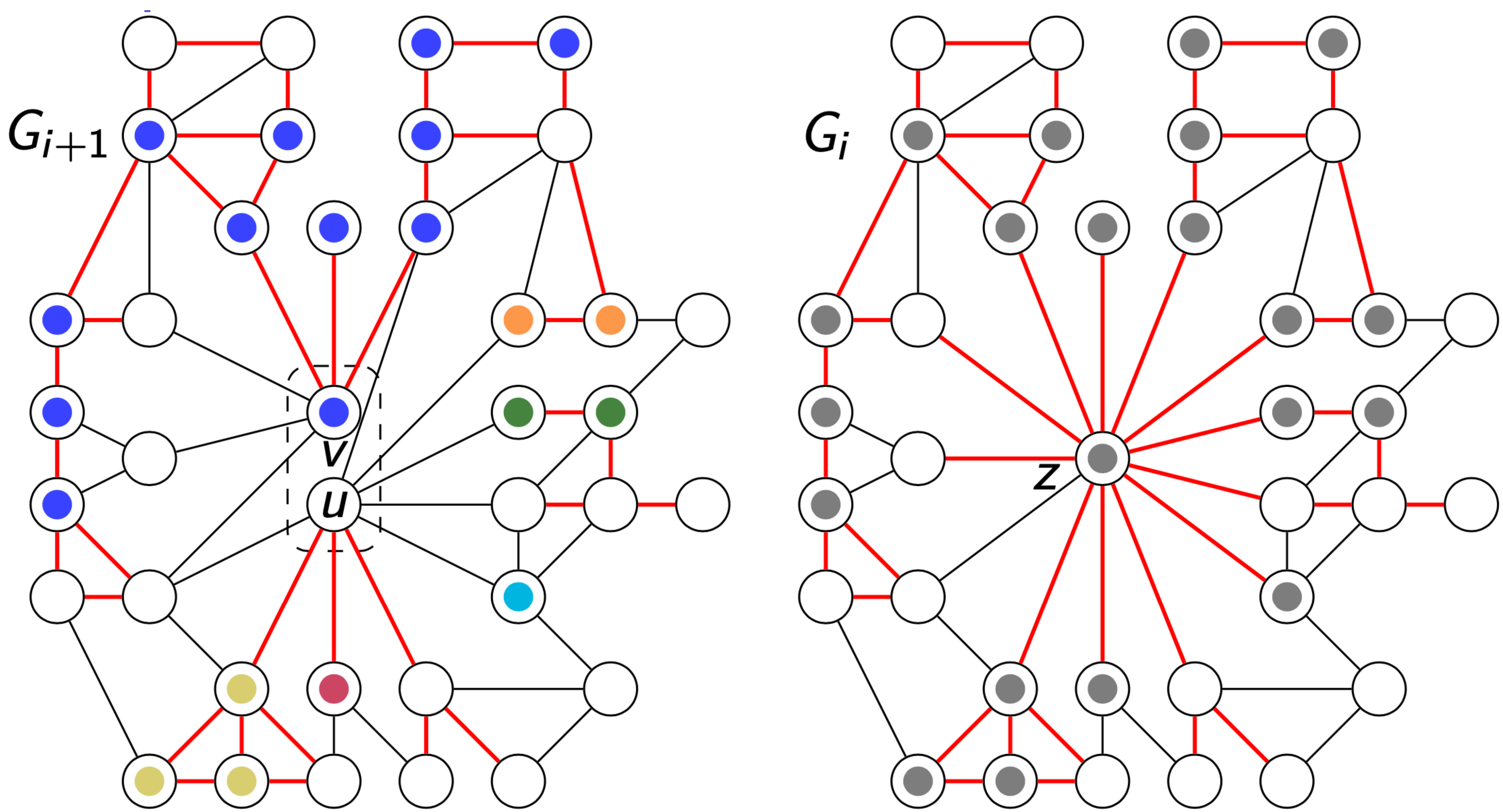
Assuming that no realizable set of size ≥ k was found so far,

➔ Best partial solution S realizing T, induces connected red components of T-z+u, T-z+v, or T-z+{u,v} of size at most k each.



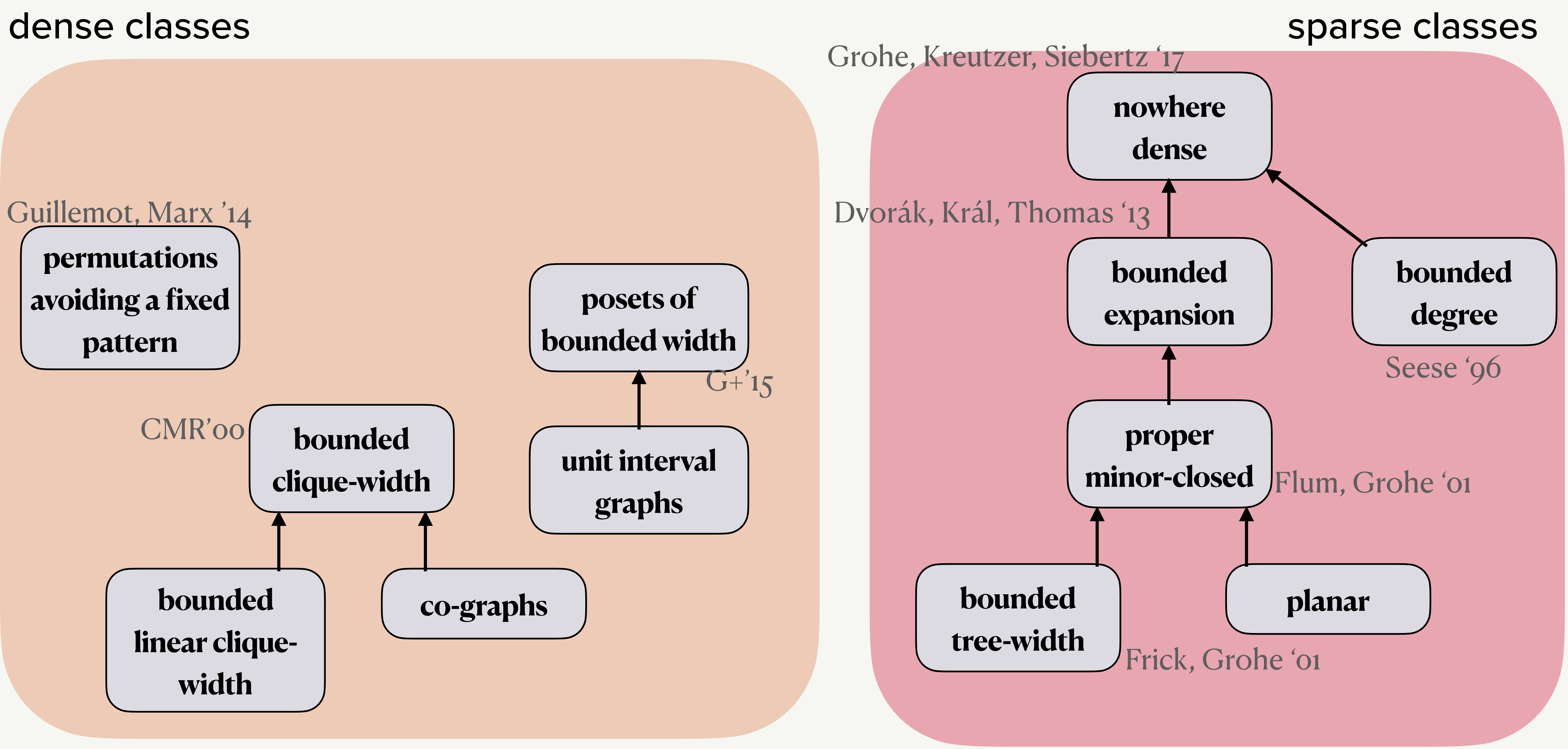
Assuming that no realizable set of size ≥ k was found so far,

➔ Best partial solution S realizing T, induces connected red components of T-z+u, T-z+v, or T-z+{u,v} of size at most k each.

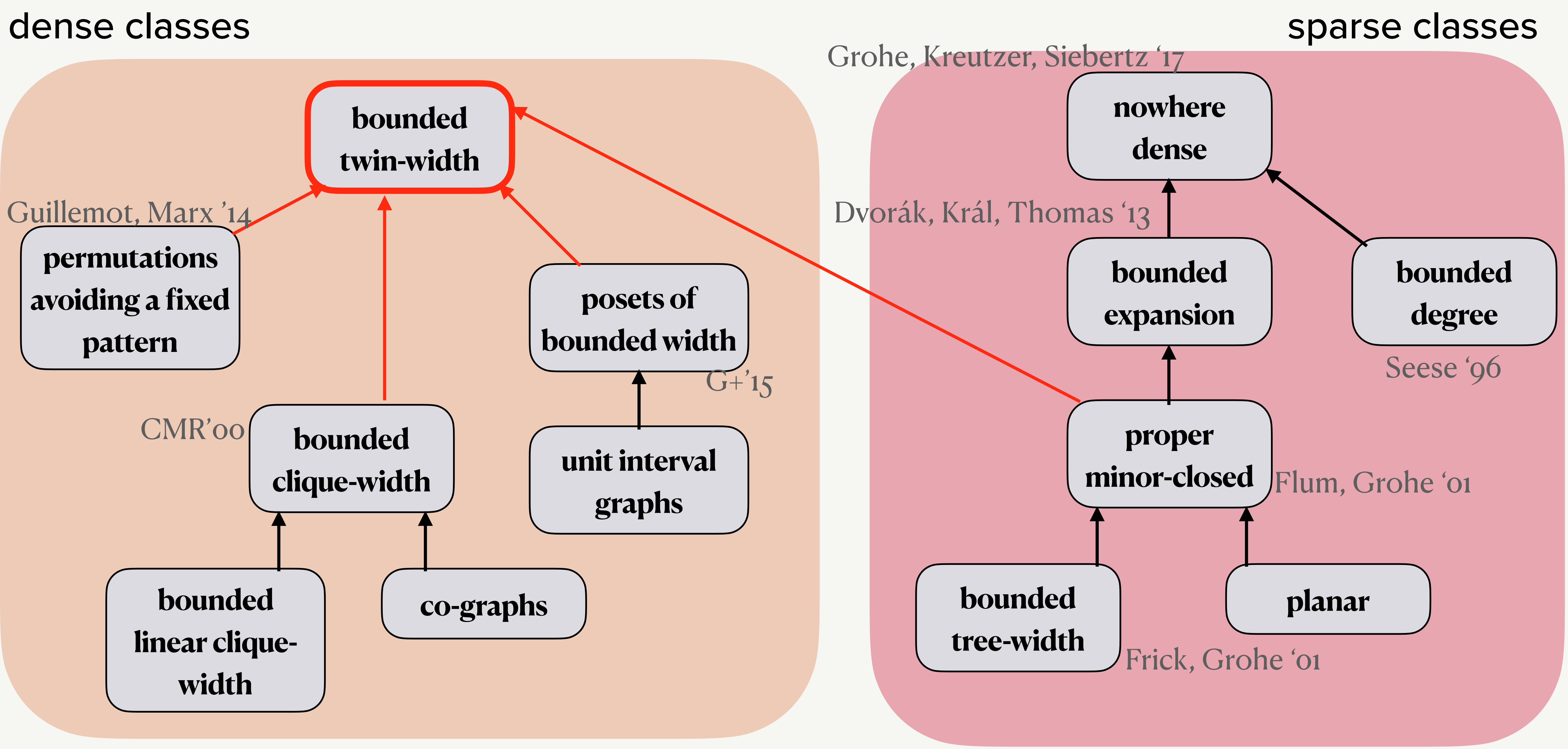


In a graph of max degree ≤ d,
there are at most $(d^{2k-2} + 1)|X|$ connected sets of size at most k
containing a set X.

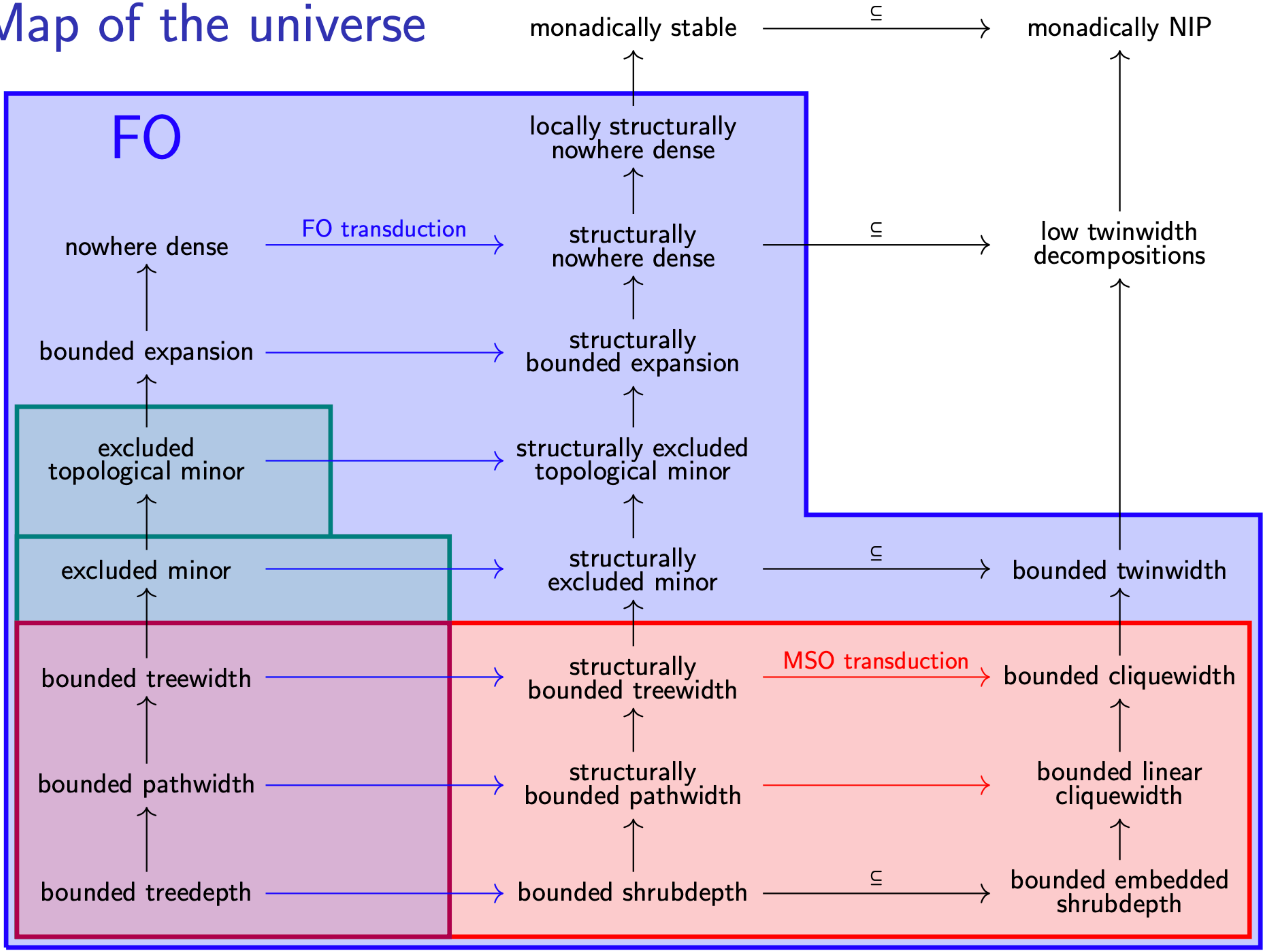
FO-model checking is FPT [BKTW'20]



FO-model checking is FPT [BKTW'20]



Map of the universe

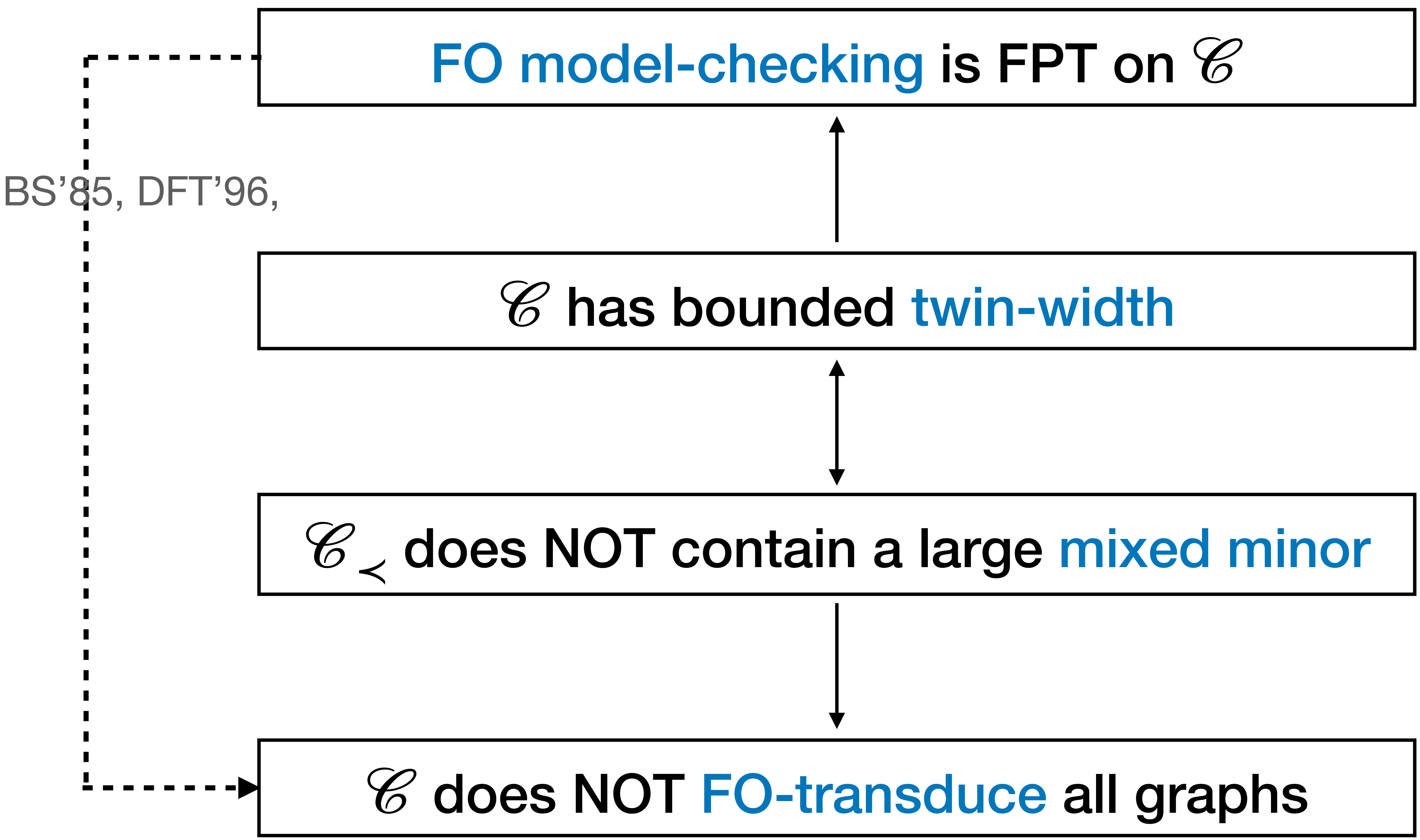


Borrowed from Sebastian Siebertz's slides

Twin

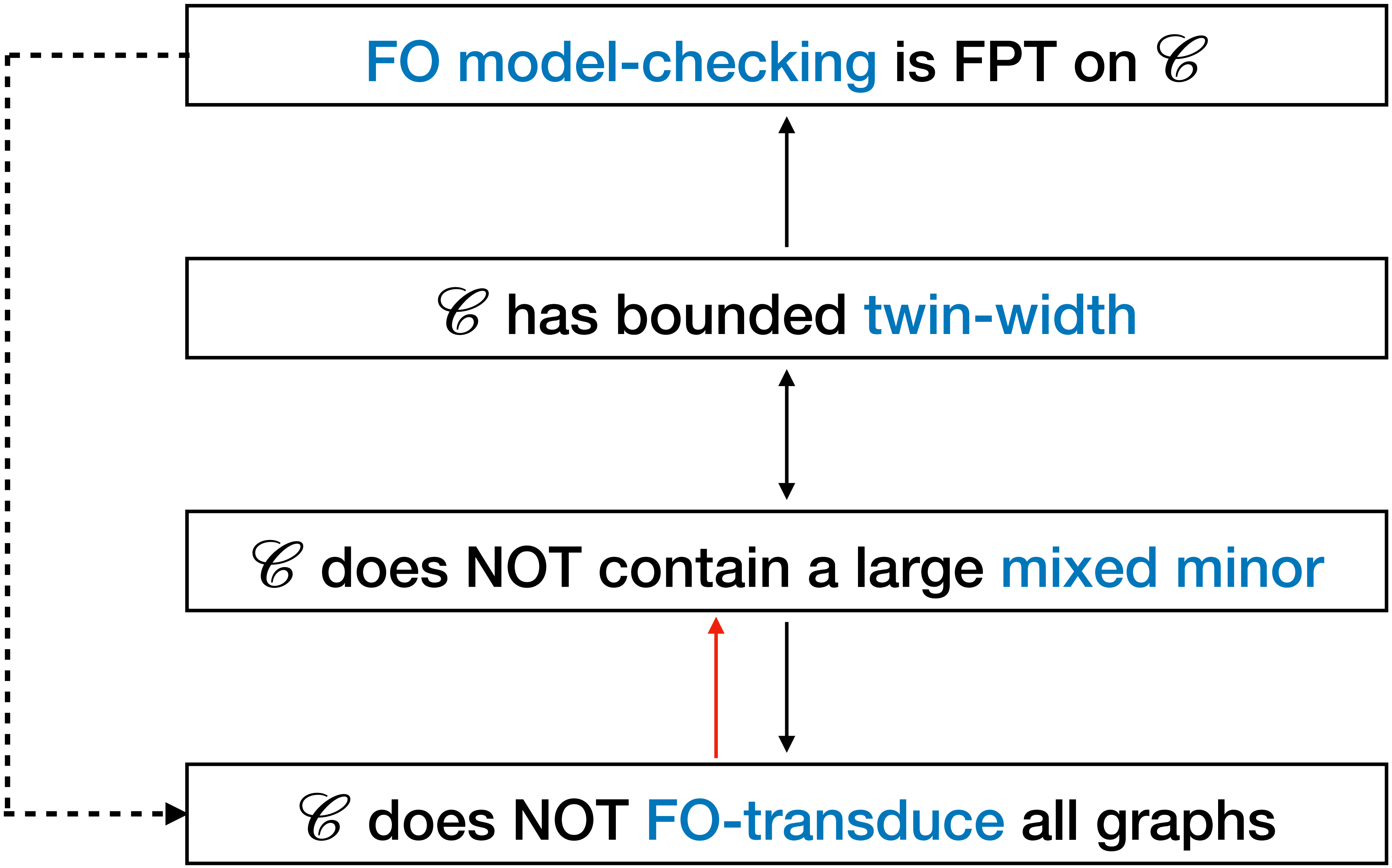
For a hereditary class $\mathscr{C}$

[Bonnet, K, Thomassé, Watrigant ’20]



Summary II

For a hereditary class $\mathscr{C}$ of **interval graphs | permutations | ordered graphs | tournaments | circle graphs | rooted directed path graphs**



Concluding Remarks

- For all the classes which are known to have bounded twin-width, we known how to compute the (approximate) contraction sequence in time $f(d) \cdot n$.

- We still do not know how to compute f(tww)-contraction sequence in FPT, even in XP time, when the input graph is arbitrary. $O(\sqrt{n} \cdot \log n)$-approximation (?) is recently obtained.

Concluding Remarks

- For all the classes which are known to have bounded twin-width, we known how to compute the (approximate) contraction sequence in time $f(d) \cdot n$.

- We still do not know how to compute f(tww)-contraction sequence in FPT, even in XP time, when the input graph is arbitrary. $O(\sqrt{n} \cdot \log n)$-approximation (?) is recently obtained.

- Characterizing the hereditary classes on which FO model checking is in FPT is a very active topic recently.
  Conjecture: FO model checking on $\mathscr{C}$ is FPT if and only if $\mathscr{C}$ does not transduce the class of all graphs (a.k.a. monadic NIP).
  Just a few weeks ago, a combinatorial characterization of monadic NIP class was announced, perhaps we're just a few steps from the conjecture to be confirmed.

Thank you!