First-order logic over classes of graphs with bounded expansion

Wojciech Kazana
INRIA Saclay, ENS de Cachan

Based on a joint work with Luc Segoufin.

"Finite and Algorithmic Model Theory 2012", Les Houches
May 14th 2012

A classical model checking problem is a question of the form:

Given a model Γ and a sentence ϕ, does $\Gamma \models \phi$?

A classical model checking problem is a question of the form:

Given a model Γ and a sentence ϕ, does $\Gamma \models \phi$?

An evaluation problem is:

Given a model Γ and a formula $\phi(\bar{x})$ find all tuples $\bar{a}$ such that $\Gamma \models \phi(\bar{a})$.

An enumeration problem is an extension of the evaluation, where we are interested in computing the same set of tuples, but the desired algorithm has two distinct phases:

- a precomputation phase building an index structure,
- followed by a phase enumerating the answers with no repetitions (the time between two consecutive outputs is called the *delay*).

In the enumeration scenario, if the precomputation phase is done in linear time in the size of the input and the delay is constant, we say that the enumeration problem belongs to class $\textsc{Constant-Delay}_{lin}$.

In the enumeration scenario, if the precomputation phase is done in linear time in the size of the input and the delay is constant, we say that the enumeration problem belongs to class $\textsc{Constant-Delay}_{lin}$.

Two additional assumptions:

- the index structure from precomputation phase is read-only during the enumeration phase,
- during the enumeration phase, only a constant amount of read-write memory is available.

Today we want to focus on two results:

Theorem (Dvořák, Král, Thomas FOCS'10 and Grohe, Kreutzer '11)

Let $\mathcal{C}$ be class of graph with bounded expansion and ϕ a FO *sentence. Given graph $G \in \mathcal{C}$ we can decide, in time linear in the size of G, whether $G \models \phi$.*

Theorem

Let $\mathcal{C}$ be class of graph with bounded expansion and $\phi(\bar{x})$ a FO *query. Given as input a graph $G \in \mathcal{C}$, the problem of enumerating all solutions to ϕ over G belongs to* $\textsc{Constant-Delay}_{lin}$.

Today we want to focus on two results:

Theorem (Dvořák, Král, Thomas FOCS'10 and Grohe, Kreutzer '11)

Let $\mathcal{C}$ be class of graph with bounded expansion and ϕ a FO *sentence. Given graph $G \in \mathcal{C}$ we can decide, in time linear in the size of G, whether $G \models \phi$.*

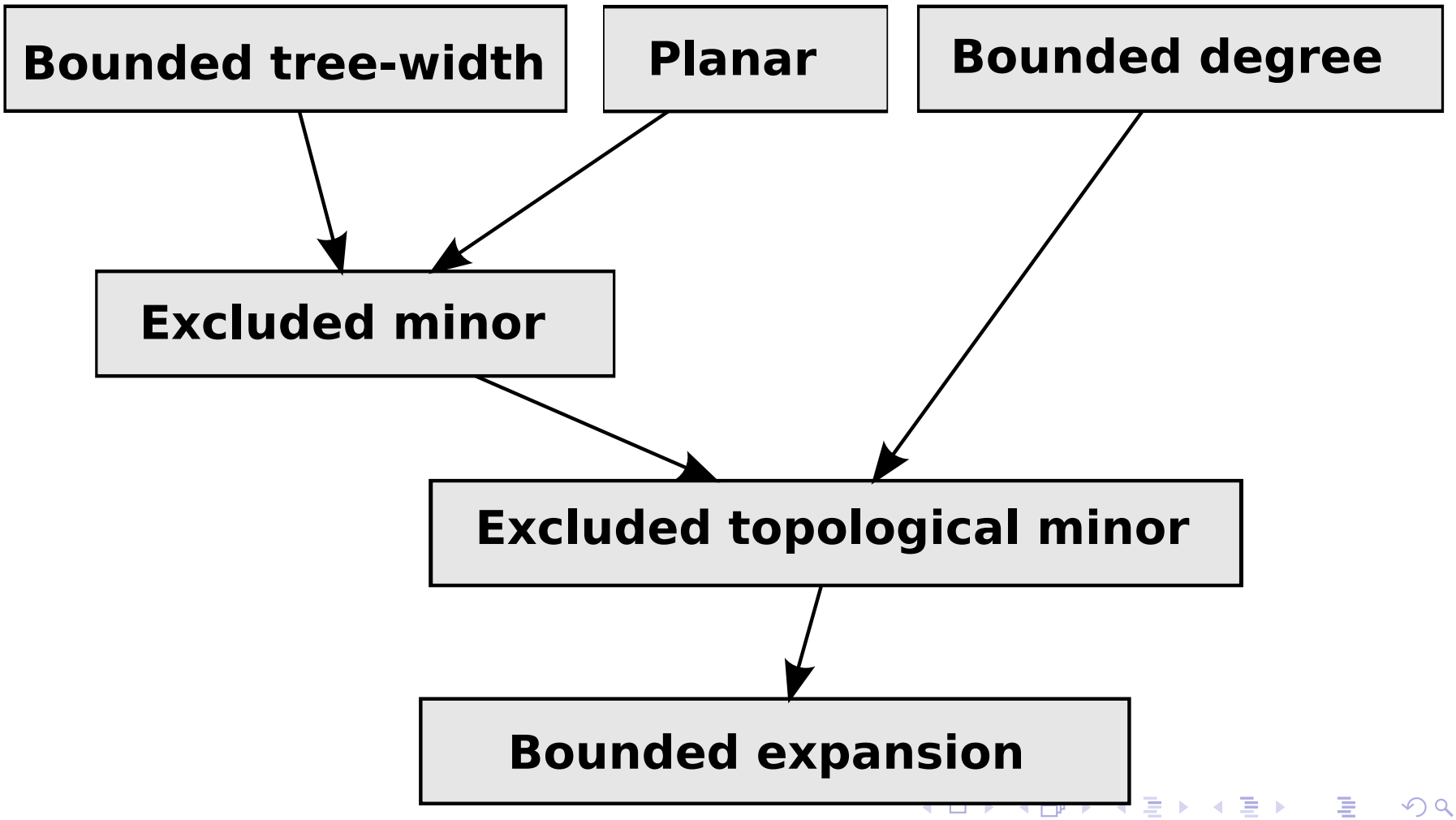
Theorem

Let $\mathcal{C}$ be class of graph with bounded expansion and $\phi(\bar{x})$ a FO *query. Given as input a graph $G \in \mathcal{C}$, the problem of enumerating all solutions to ϕ over G belongs to* $\textsc{Constant-Delay}_{lin}$.

Both theorems hold for classes of relational structures with bounded expansion.

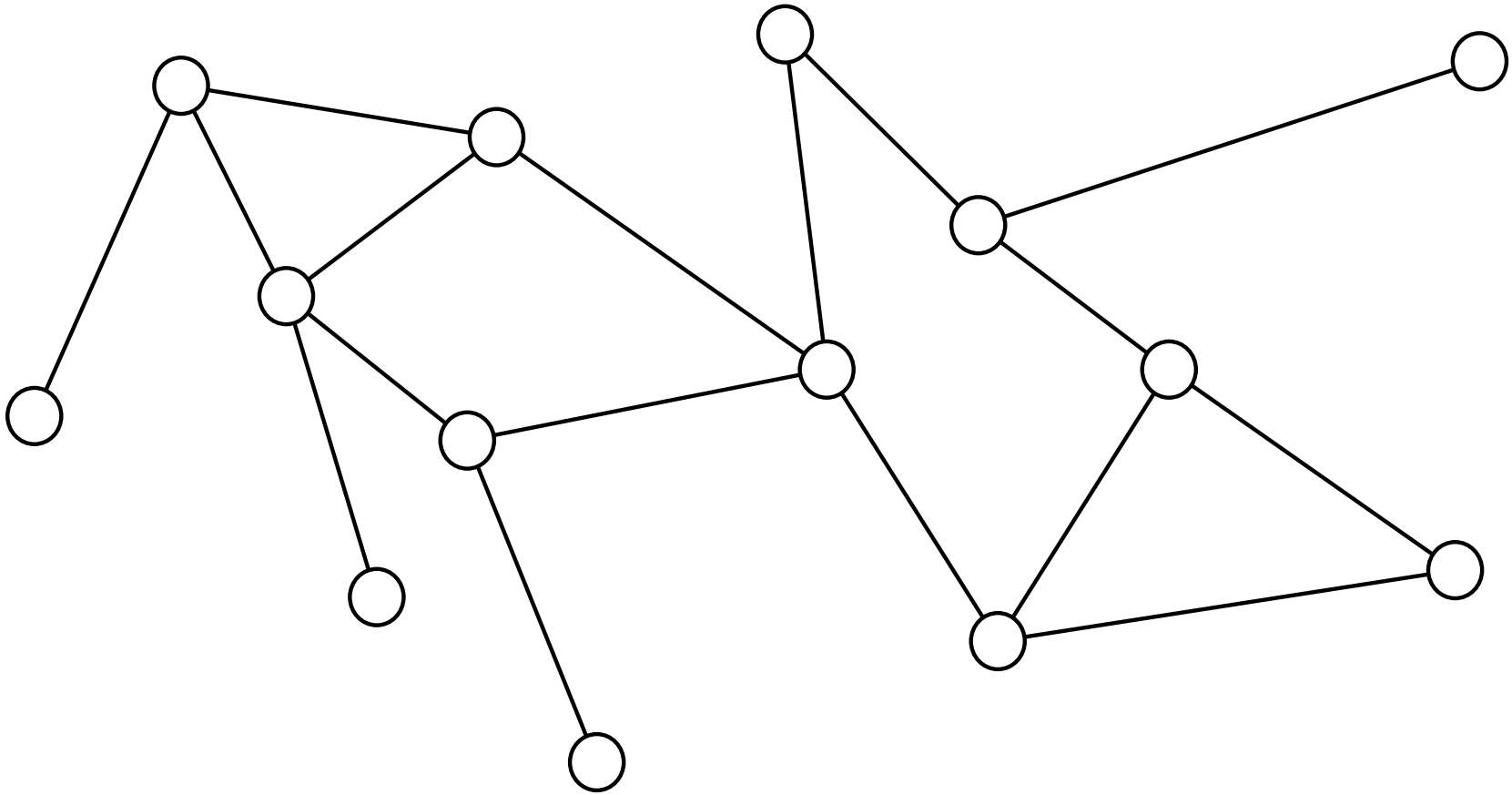
Why bounded expansion?

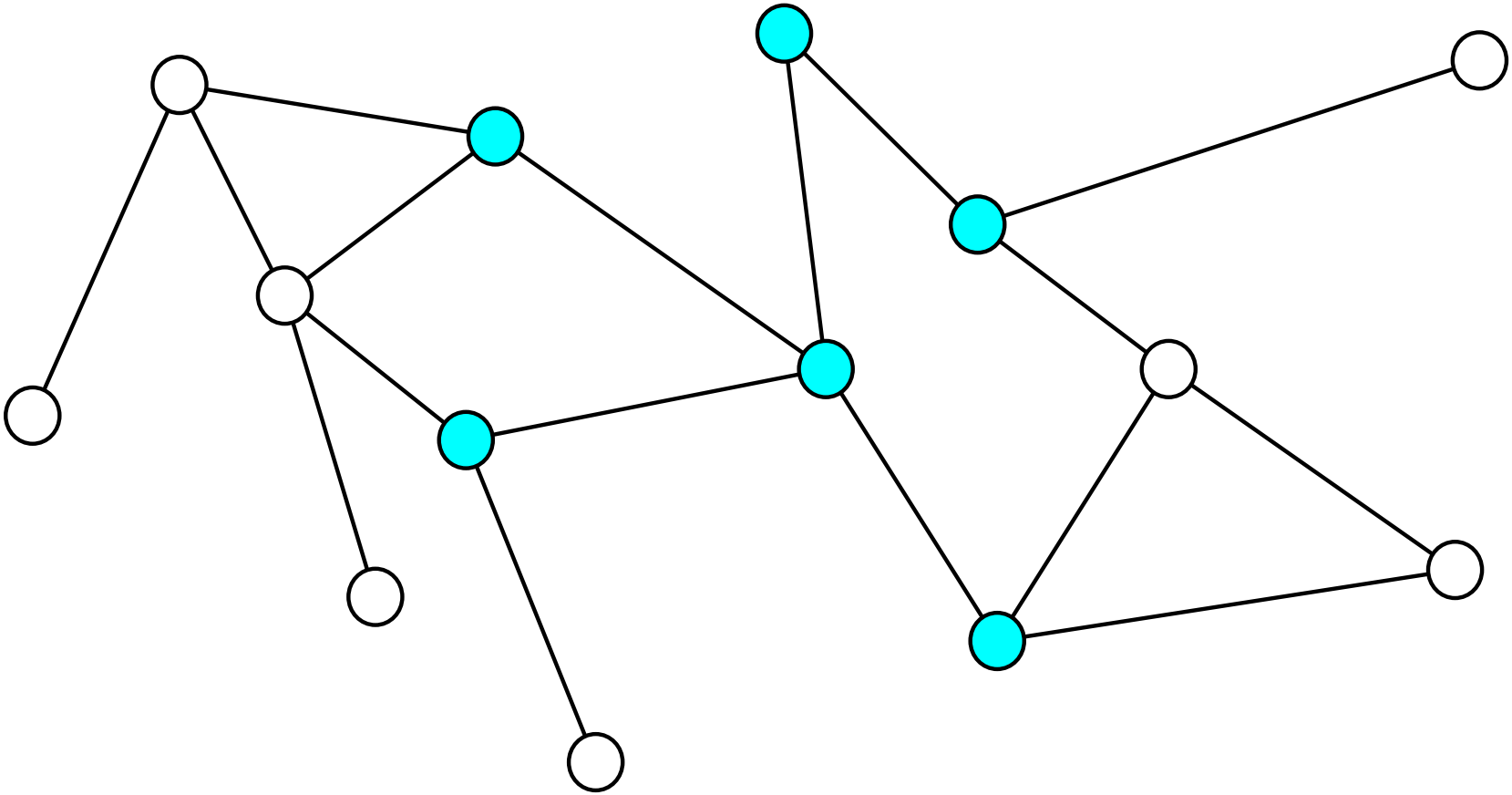
Why bounded expansion?

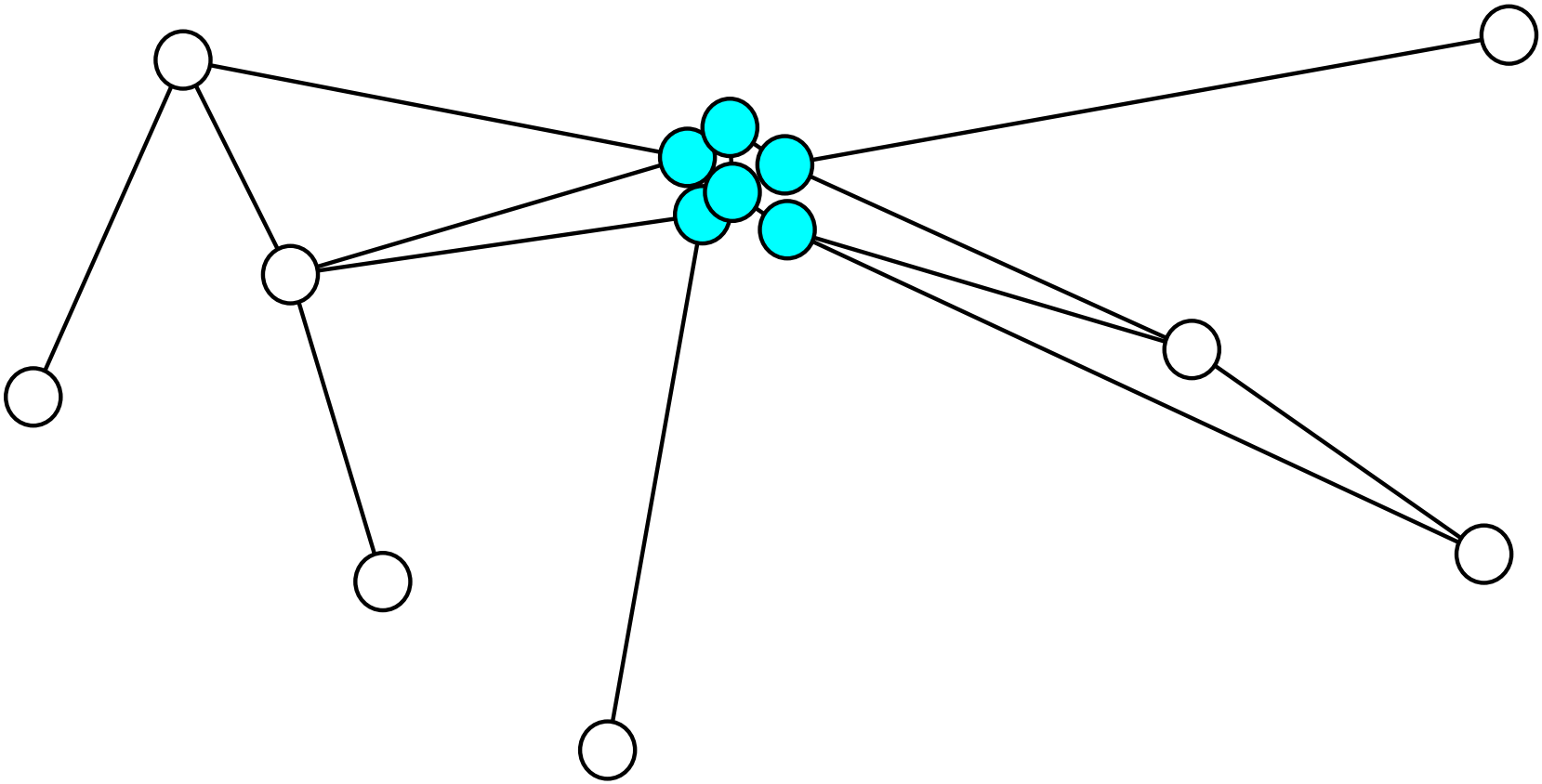


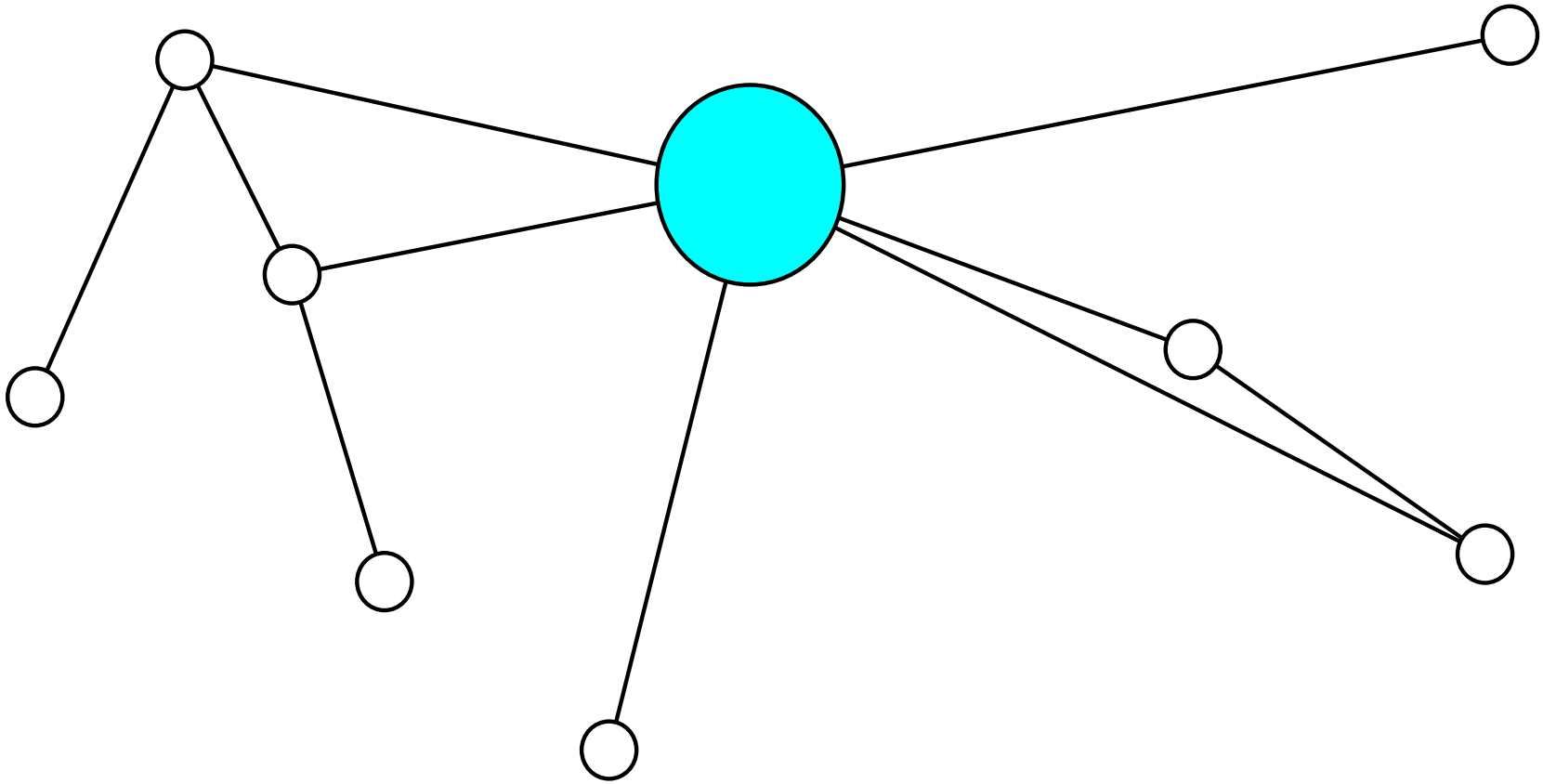
Some necessary definitions:

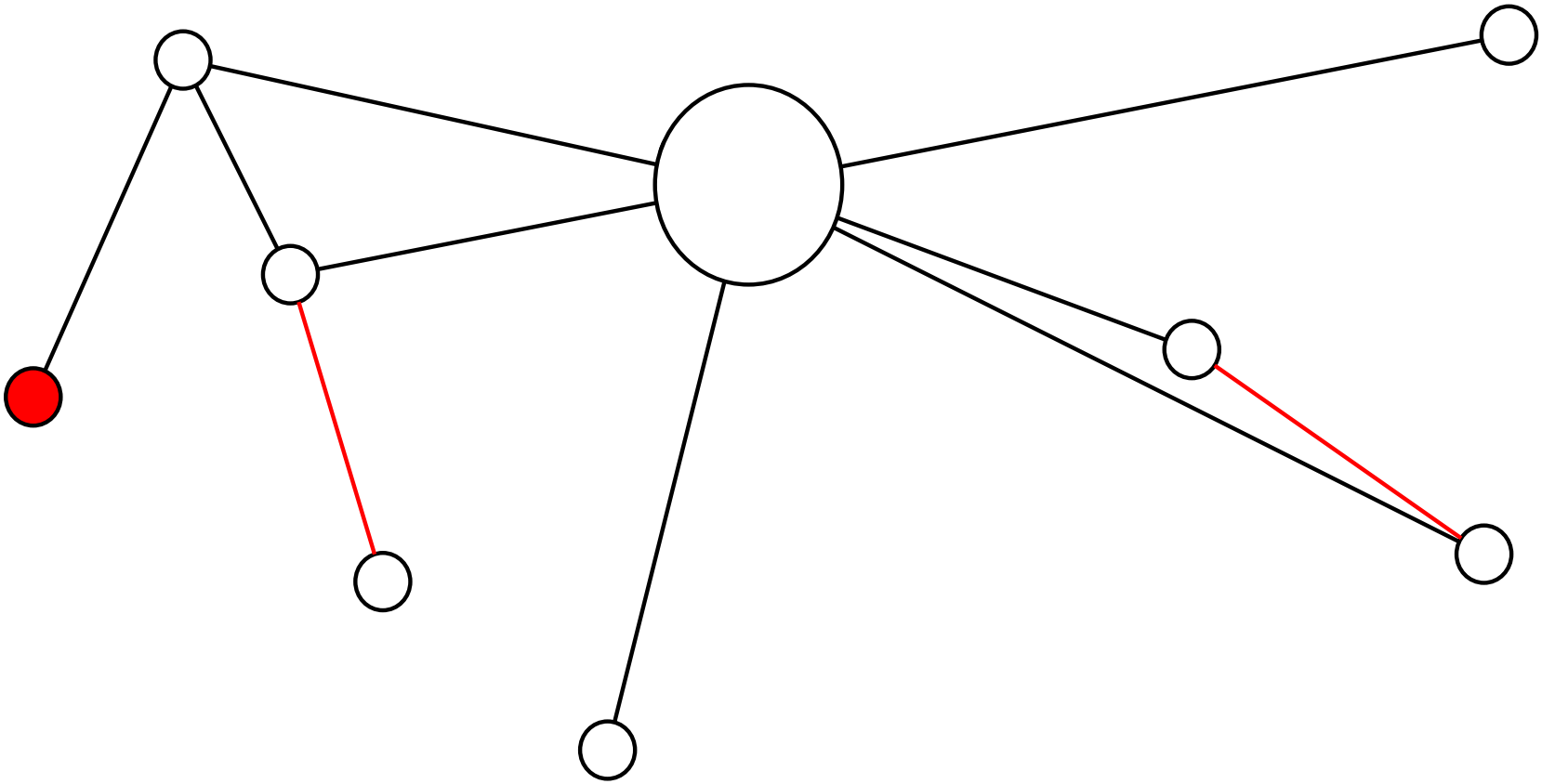
- $G = (V, E)$ is a graph with V representing the set of nodes and $E \subseteq V^2$ the set of edges.
- Given graph G and $r \in \mathbb{N}$, we say that H is an r-minor of G if H is obtained from G by *collapsing* pairwise disjoint subgraphs $V_1, \ldots, V_k$ of radius $\leq r$ and removing any edges and nodes from the obtained subgraph.
- We denote the set of r-minors with $G\nabla_r$ (so $H \in G\nabla_r$ means that H is r-minor of G).
- For example $G\nabla_0$ is the set of all subgraphs of G.

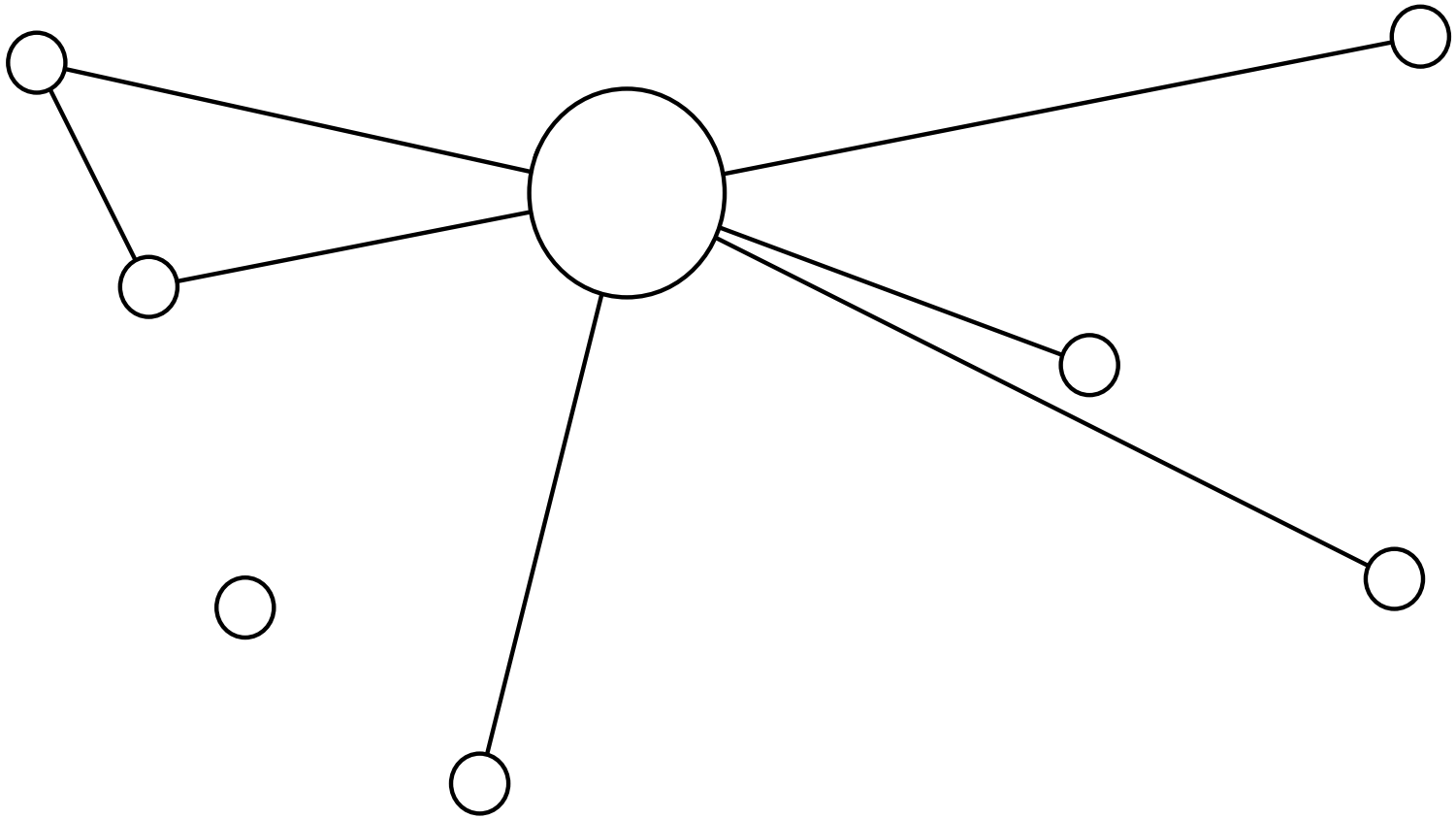












One more definition:

- For a graph G, the *greatest reduced average density (grad) of G with rank r* is:

$$\nabla_r(G) = max_{H \in G \nabla_r} \frac{||H||}{|H|},$$

where $||H||$ is the number of edges in H and $|H|$ is the number of nodes in H.

Definition

Let $\mathcal{C}$ be a (possibly infinite) class of graphs. We say that $\mathcal{C}$ has bounded expansion if there exists a computable function $f : \mathbb{N} \to \mathbb{R}$ such that for all graphs $G \in \mathcal{C}$ and for all $r \in \mathbb{N}$ we have:

$$\nabla_r(G) \leq f(r).$$

This notion is actually very robust:

Theorem (Nesetril, Ossona de Mendez)

Let $\mathcal{C}$ be a class of graphs. Then the following are equivalent:

- *$\mathcal{C}$ has bounded expansion,*
- *$\mathcal{C}$ has low tree-width colorings,*
- *$\mathcal{C}$ has low tree-depth colorings,*
- *$\mathcal{C}$ has p-centered colorings,*
- *there exists a function $f : \mathbb{N} \to \mathbb{R}$ such that each graph $G \in \mathcal{C}$ has an orientation $\vec{G}$ such that the maximal in-degree of a node in $\vec{G}$ (denoted $\Delta^-(\vec{G})$) is bounded by $f(1)$ and for each such orientation $\vec{G}$ there exists a transitive fraternal augmentation $\vec{G} = \vec{G}_1 \subseteq \vec{G}_2 \subseteq \ldots \vec{G}_i \subseteq \vec{G}_{i+1} \subseteq \ldots$ such that for each $i \geq 1$ we have $\Delta^-(\vec{G}_i) \leq f(i)$.*

The proof of Theorem 1 by Dvořák, Král, Thomas and by Grohe, Kreutzer use the "low tree-depth colorings" definition.

In this talk we shall exploit the notion of transitive fraternal augmentations.

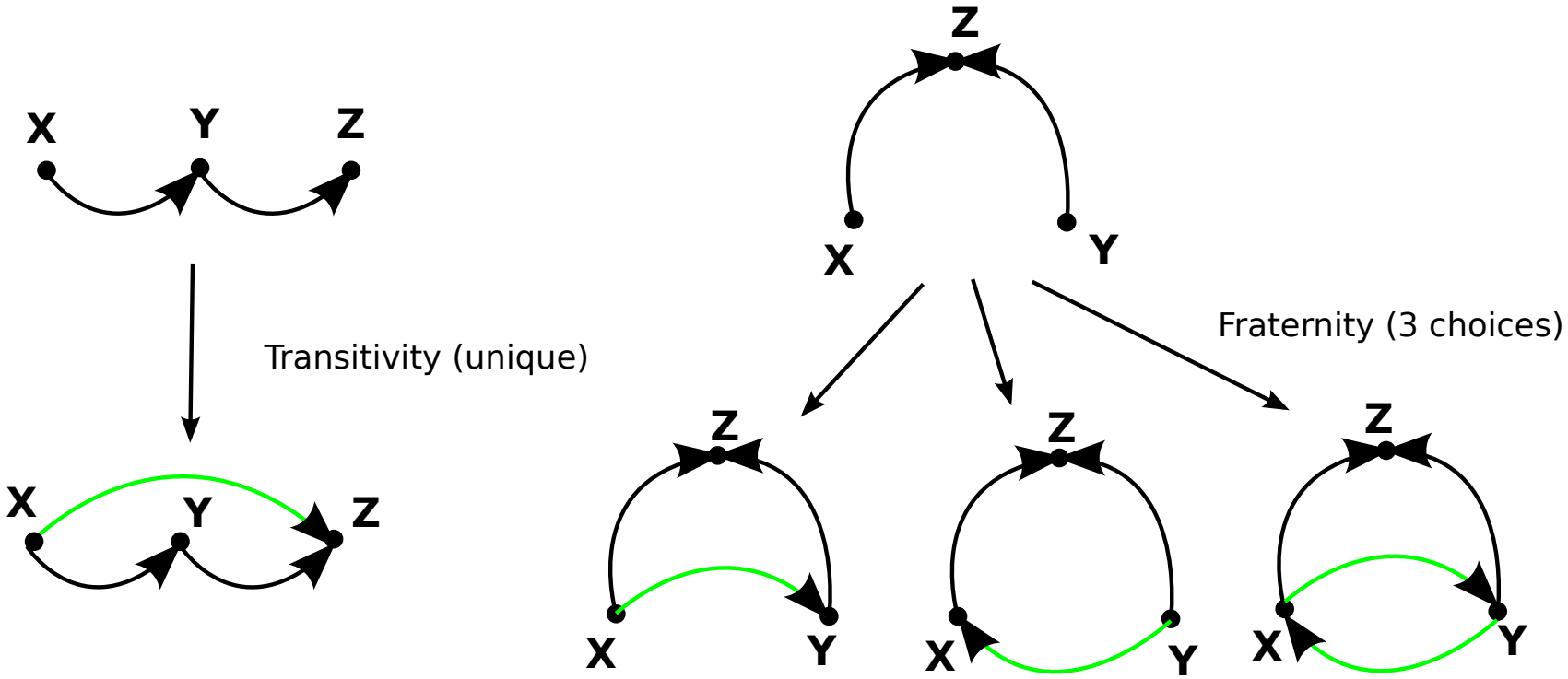
Definition

Let $\vec{G}$ be a directed graph. A 1-transitive fraternal augmentation of $\vec{G}$ is any graph $\vec{H}$ with the same vertex set as $\vec{G}$, including all edges of $\vec{G}$ (with their orientation) and such that for any three vertices x, y, z of $\vec{G}$ we have the following:

- *if (x, y) and (y, z) are edges in $\vec{G}$, then (x, z) is an edge in $\vec{H}$ (transitivity),*
- *if (x, z) and (y, z) are edges in $\vec{G}$, then at least one of the edges: (x, y), (y, x) is in $\vec{H}$ (fraternity).*

Note that the notion of 1-transitive fraternal augmentation is not a fixed operation. Although transitivity induces precise edges, fraternity implies nondeterminism and thus there can possibly be many different 1-transitive fraternal augmentations.

Original graph G



Possible 1-transitive fraternal augmentations.

Getting back to the definition:

Definition

There exists a function $f : \mathbb{N} \to \mathbb{R}$ such that each graph $G \in \mathcal{C}$ has an orientation $\vec{G}$ such that the maximal in-degree of a node in $\vec{G}$ (denoted $\Delta^-(\vec{G})$) is bounded by $f(1)$ and for each such orientation $\vec{G}$ there exists a transitive-fraternal augmentation $\vec{G} = \vec{G}_1 \subseteq \vec{G}_2 \subseteq \ldots \vec{G}_i \subseteq \vec{G}_{i+1} \subseteq \ldots$ such that for each $i \geq 1$ we have $\Delta^-(\vec{G}_i) \leq f(i)$.

We can view that as: we start with a graph $G \in \mathcal{C}$. We find its orientation $\vec{G}$ such that $\Delta^-(\vec{G}) \leq f(1)$. As we said before, 1-transitive fraternal augmentation is not a fixed process. The theorem of Nesetril and Ossona de Mendez states that for classes of graphs with bounded expansion, wise choice of orientations of edges induced by fraternity relation allows us to keep control over the evolution of Δ^- during the consecutive augmentation steps.

What is important:

- for each i the value of $f(i)$ is independent of the size of G and thus is a constant in our setting,
- each $\vec{G}_i$ has size linear in the size of G and starting from $\vec{G}_i$, we can compute $\vec{G}_{i+1}$ in linear time.

Theorem

Let $\mathcal{C}$ be class of graph with bounded expansion and ϕ a FO *sentence. Given graph $G \in \mathcal{C}$ we can decide, in time linear in the size of G, whether $G \models \phi$.*

Outline of the proof:

- Use Gaifman Theorem to reduce ϕ to local formulas.
- Use augmentations to get simple description of neighborhoods.
- Use the above description to show a quantifier elimination procedure for local formulas.
- Trivial recoloring with quantifier free unary formulas.

Step (1/4): Apply Gaifman Theorem.

Theorem (Gaifman Theorem)

Every FO *formula* $\phi(\bar{x})$ *is equivalent to a Boolean combination of the following:*

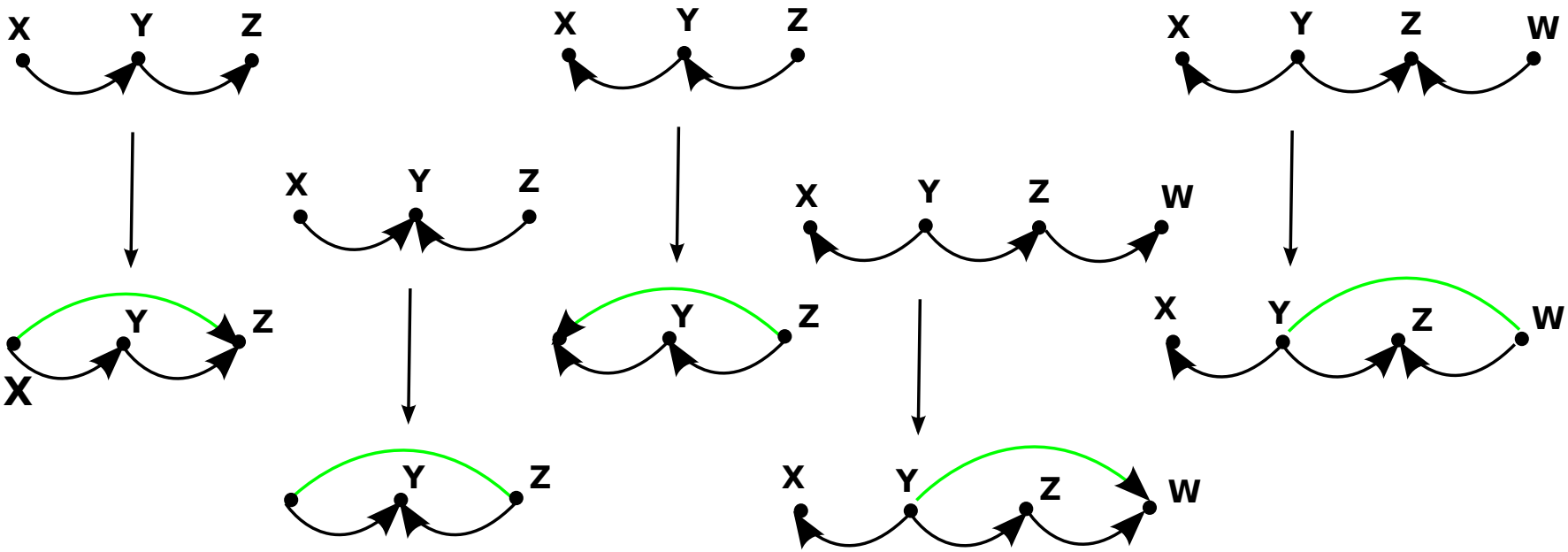
- *formulas* $\psi(\bar{x})$ *r-local around* $\bar{x}$,
- *sentences of the form* $\exists x_1, \ldots, x_s \left(\bigwedge_{i \in \{1,\ldots,s\}} \alpha^{(r)}(x_i) \wedge \bigwedge_{1 \leq i < j \leq s} \delta(x_i, x_j) > 2r \right)$, *where* $\alpha^{(r)}(x)$ *is r-local around* x.

Furthermore, the transformation from ϕ *to such a Boolean combination is effective.*

Note: r-local around x means that we quantify over nodes that are in the distance $\leq r$ from x.

Step (2/4): Augmentations vs neighborhoods.

Initial graph

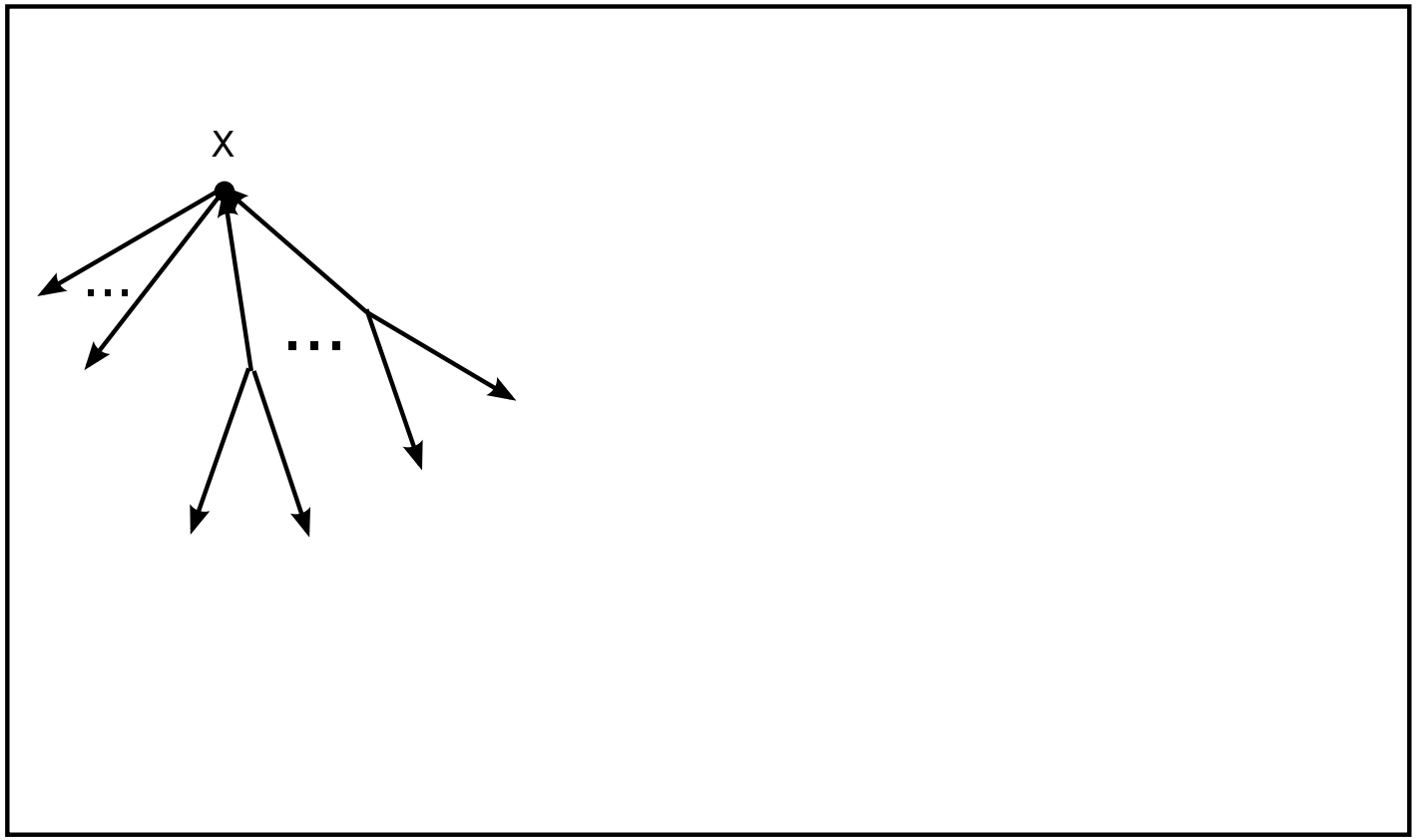


Any of its 1-transitive fraternal augmentations

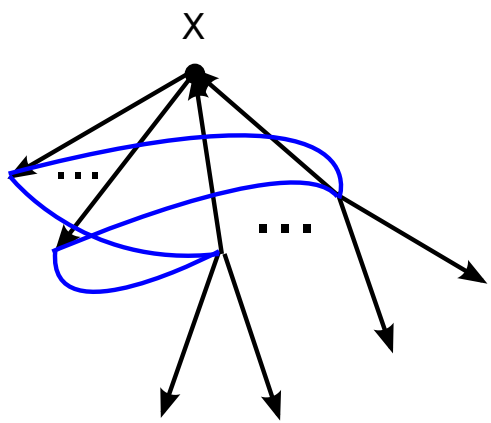
Step (2/4): Augmentations vs neighborhoods.

From the previous figure we see that after each 1-transitive fraternal augmentation every path of length 3 has length reduced to 2.
In general, after r augmentations, the initial r-neighborhood of any node x is contained in its 2-neighborhood and there is just one orientation to make distance 2 possible.

Step (2/4): Augmentations vs neighborhoods.

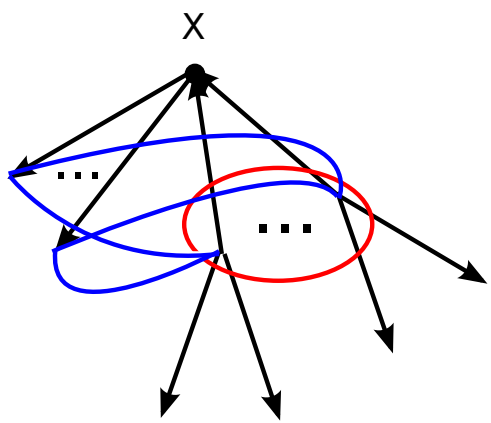


Step (2/4): Augmentations vs neighborhoods.



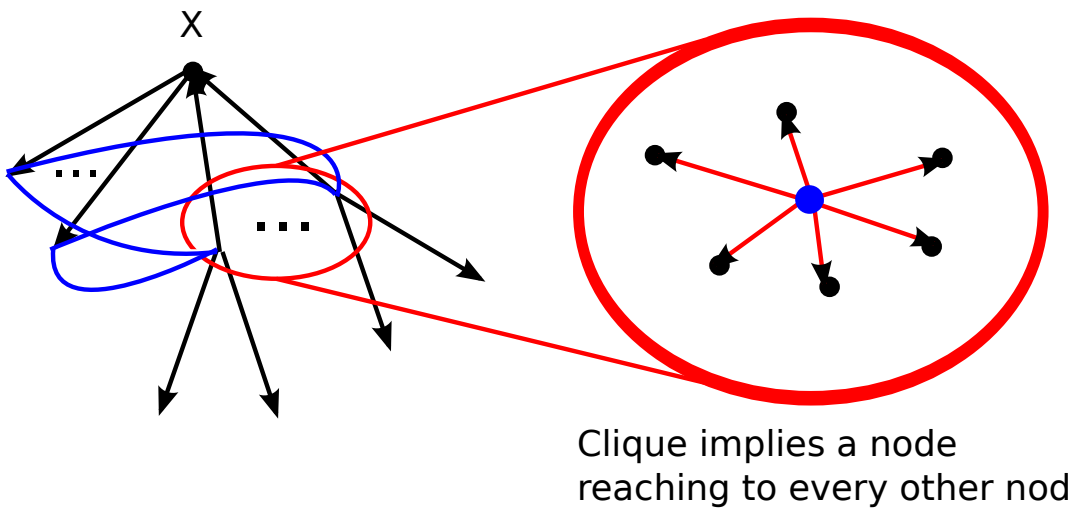
Transitivity

Step (2/4): Augmentations vs neighborhoods.



Fraternity
Transitivity

Step (2/4): Augmentations vs neighborhoods.

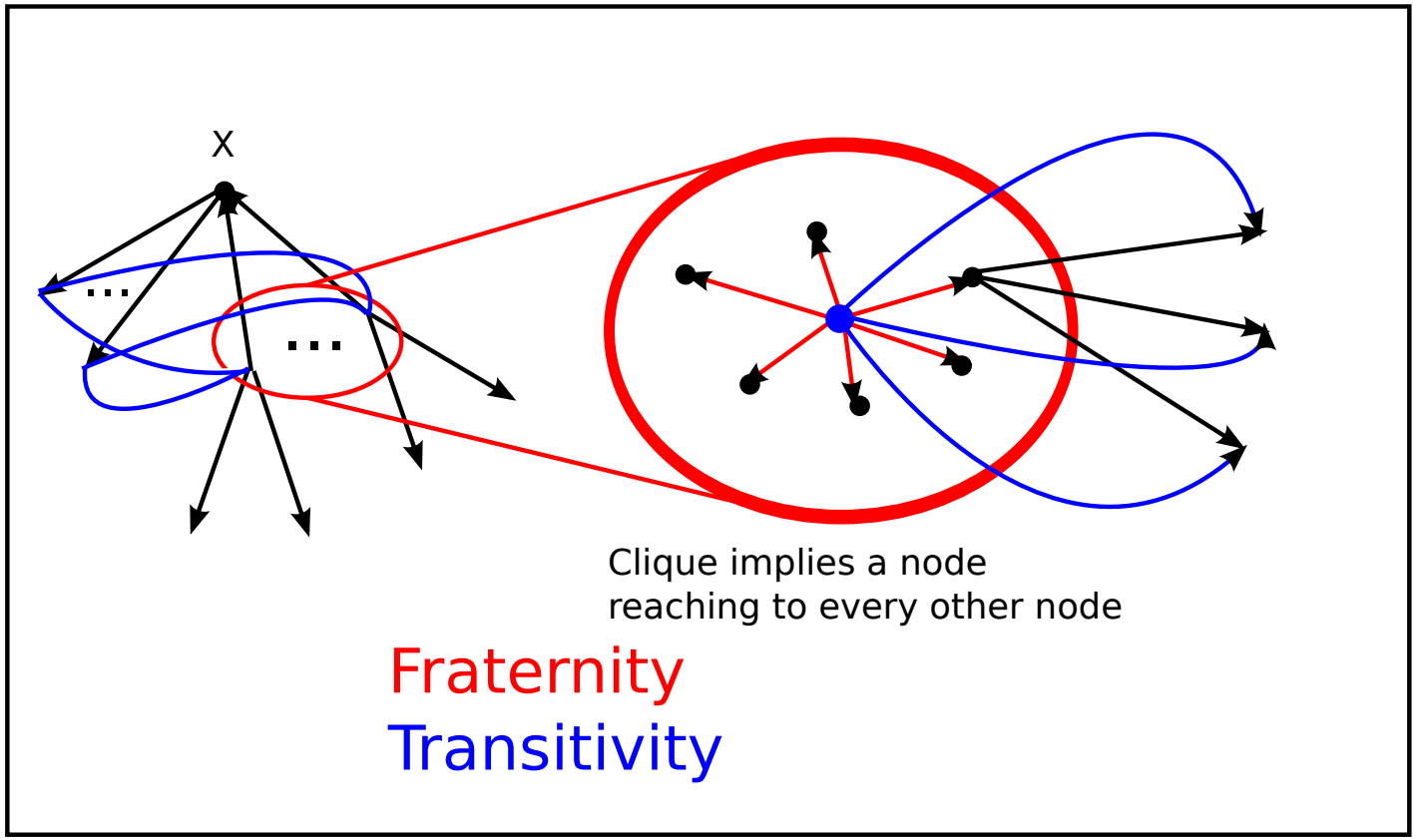


Clique implies a node
reaching to every other node

Fraternity

Transitivity

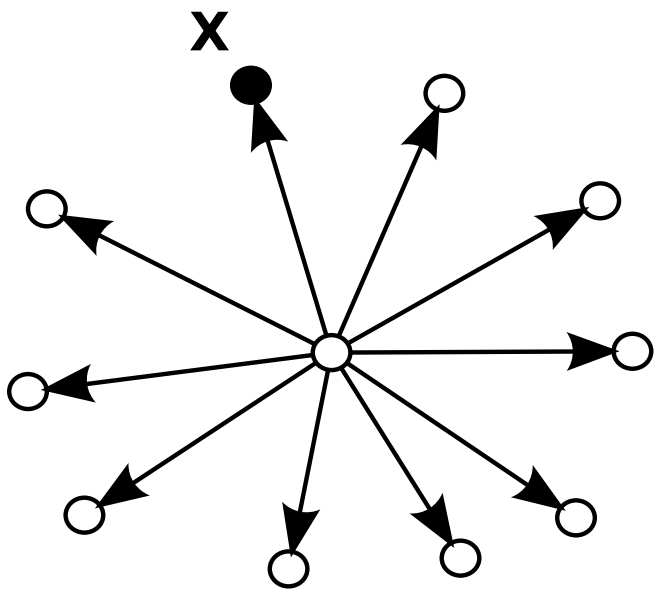
Step (2/4): Augmentations vs neighborhoods.



Step (2/4): Augmentations vs neighborhoods.

From the previous figures we see that after a fixed number of 1-transitive fraternal augmentations, every node x is backward-connected to a node x_c (called *the real center*) such that every node from the initial r-neighborhood of x is now directly reachable from x_c.

Step (2/4): Augmentations vs neighborhoods.



Initial r-neighborhood of X.

Step (3/4): Quantifier elimination.

Recall that each node in the graph (at any fixed phase of the augmentation process) has a bounded number of backward connections.

Step (3/4): Quantifier elimination.

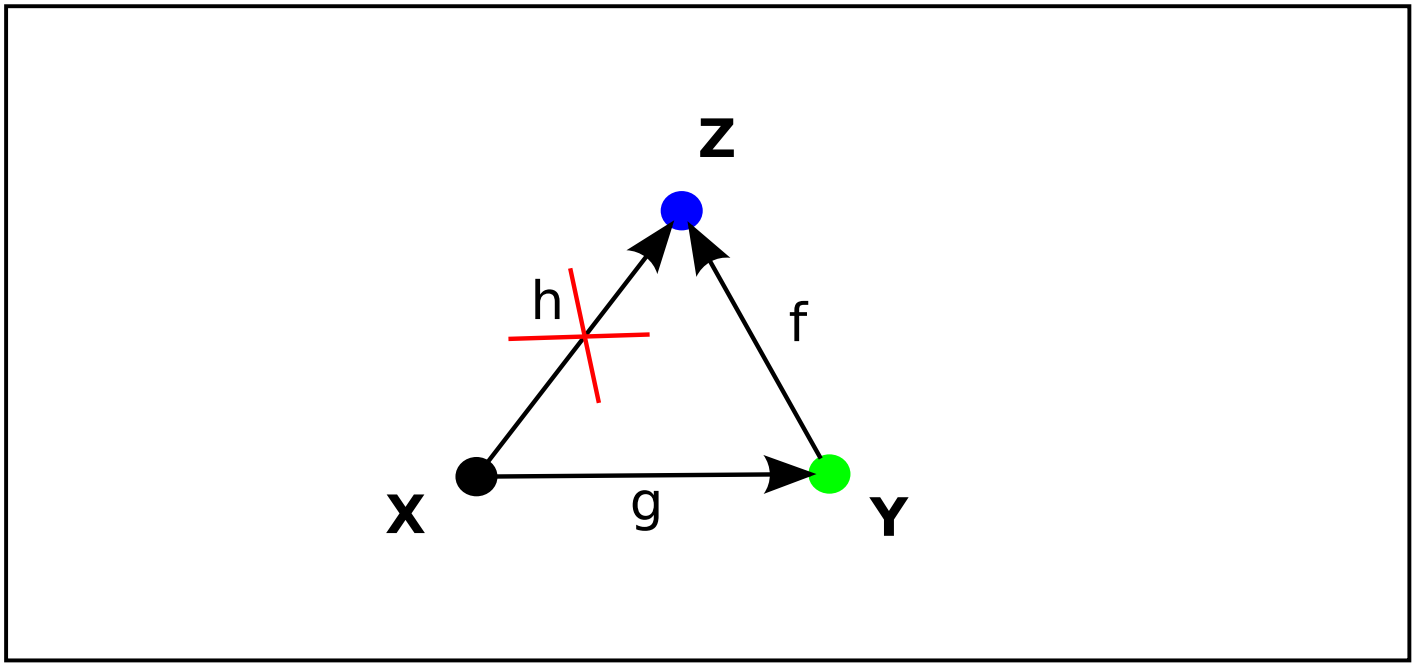
Recall that each node in the graph (at any fixed phase of the augmentation process) has a bounded number of backward connections.
If this bound is Δ^-, then Δ^- functions $f, g, h, \ldots$ is enough to describe edges of the graph.

Step (3/4): Quantifier elimination.

Let us consider the following example:

$$\phi(x,y) = \exists_z f(z) = y \wedge g(y) =$$
$$x \wedge \mathrm{Black}(x) \wedge \mathrm{Green}(y) \wedge \mathrm{Blue}(z) \wedge \neg h(z) = x.$$

Step (3/4): Quantifier elimination.
Let us consider the following example:
$\phi(x, y) = \exists_z f(z) = y \wedge g(y) =$
$x \wedge \text{Black}(x) \wedge \text{Green}(y) \wedge \text{Blue}(z) \wedge \neg h(z) = x.$



Step (3/4): Quantifier elimination.

Instead of just verifying if the above z exists, we do a bit more: we count the number of valid candidates for z.
We do this for:

- $\psi_1(x,y) = \exists_z f(z) = y \wedge g(y) = x \wedge \mathrm{Black}(x) \wedge \mathrm{Green}(y) \wedge \mathrm{Blue}(z)$,
- $\psi_2(x,y) = \exists_z f(z) = y \wedge g(y) = x \wedge \mathrm{Black}(x) \wedge \mathrm{Green}(y) \wedge \mathrm{Blue}(z) \wedge h(z) = x$.

Step (3/4): Quantifier elimination.

The number $\#_\phi(x, y)$ of valid z-s is then equal to:
$\#_\phi(x, y) = \#_{\psi_1}(x, y) - \#_{\psi_2}(x, y)$.

Step (3/4): Quantifier elimination.

The number $\#_\phi(x, y)$ of valid z-s is then equal to:
$\#_\phi(x, y) = \#_{\psi_1}(x, y) - \#_{\psi_2}(x, y)$.
But it turns out that $\#_{\psi_1}(x, y)$ and $\#_{\psi_2}(x, y)$ in fact depend only on y (x is uniquely determined by y).

Step (3/4): Quantifier elimination.

The number $\#_\phi(x, y)$ of valid z-s is then equal to:
$\#_\phi(x, y) = \#_{\psi_1}(x, y) - \#_{\psi_2}(x, y)$.
But it turns out that $\#_{\psi_1}(x, y)$ and $\#_{\psi_2}(x, y)$ in fact depend only on y (x is uniquely determined by y).
So each node v in the graph can store values of $\#_{\psi_1}(x, y)$ and $\#_{\psi_2}(x, y)$ in case it would be assigned for y.

Step (3/4): Quantifier elimination.

Using the above observation we see that:
$\phi(x,y) = \exists_z f(z) = y \wedge g(y) =$
$x \wedge \mathrm{Black}(x) \wedge \mathrm{Green}(y) \wedge \mathrm{Blue}(z) \wedge \neg h(z) = x$
is equivalent to:
$\phi'(x,y) = g(y) = x \wedge \mathrm{Black}(x) \wedge \mathrm{Green}(y) \wedge \mathrm{Recolored}(y)$,
where new color "Recolored" holds for the nodes for which the value of $\#_{\psi_1}(x,y) - \#_{\psi_2}(x,y)$ was positive.

Step (4/4): Recoloring the graph.

We can inductively (starting from the most inner one) remove quantifiers from $\alpha^{(r)}(x)$ from the application of the Gaifman Theorem and obtain an equivalent $\beta^{(r)}(x)$ quantifier free formula.

Step (4/4): Recoloring the graph.

We can inductively (starting from the most inner one) remove quantifiers from $\alpha^{(r)}(x)$ from the application of the Gaifman Theorem and obtain an equivalent $\beta^{(r)}(x)$ quantifier free formula.
Note that:
The size of $\beta^{(r)}$ depends only on the quantifier depth of $\alpha^{(r)}$ and the class $\mathcal{C}$ and thus is constant in the size of the input graph.

Step (4/4): Recoloring the graph.

We can inductively (starting from the most inner one) remove quantifiers from $\alpha^{(r)}(x)$ from the application of the Gaifman Theorem and obtain an equivalent $\beta^{(r)}(x)$ quantifier free formula.
Note that:
The size of $\beta^{(r)}$ depends only on the quantifier depth of $\alpha^{(r)}$ and the class $\mathcal{C}$ and thus is constant in the size of the input graph.
We then check nodes of the input graph one by one and recolor them with color $\beta^{(r)}$ if necessary.

We now move to the second theorem:

Theorem

Let $\mathcal{C}$ be class of graph with bounded expansion and $\phi(\bar{x})$ a FO *query. Given as input a graph $G \in \mathcal{C}$, the problem of enumerating all solutions to ϕ over G belongs to* $\textsc{Constant-Delay}_{lin}$.

Outline of the proof is almost exactly the same as previously:

- Use Gaifman Theorem to reduce ϕ to local formulas.
- Use augmentations to get simple description of neighborhoods.
- Use the above description to show a quantifier elimination procedure for local formulas.
- Enumerate solutions to quantifier free local formulas.

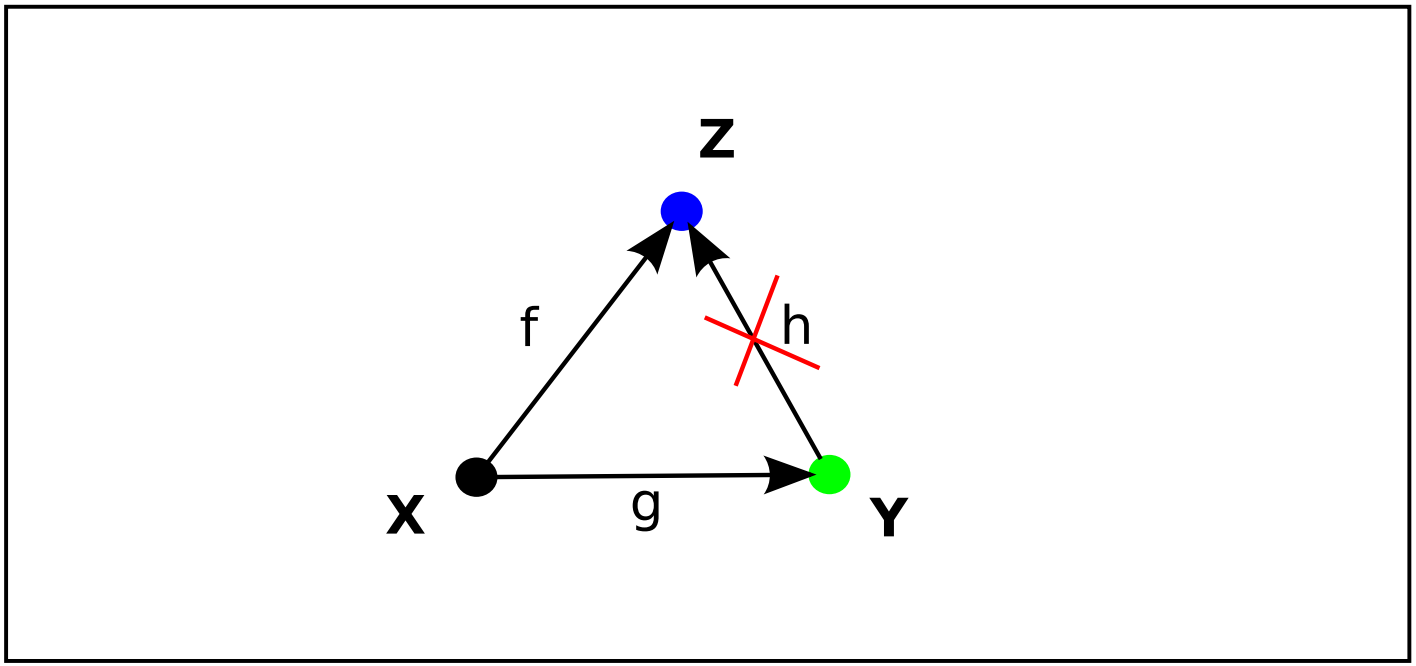
As the first three steps are similar to the previous case, we just focus on the last part.

Step (4/4): Enumerating solutions.

Let us consider the following example:
$\phi(x, y, z) = f(z) = x \wedge g(y) =$
$x \wedge \text{Black}(x) \wedge \text{Green}(y) \wedge \text{Blue}(z) \wedge \neg h(z) = y.$

Step (4/4): Enumerating solutions.

Let us consider the following example:
$\phi(x, y, z) = f(z) = x \wedge g(y) =$
$x \wedge \text{Black}(x) \wedge \text{Green}(y) \wedge \text{Blue}(z) \wedge \neg h(z) = y.$

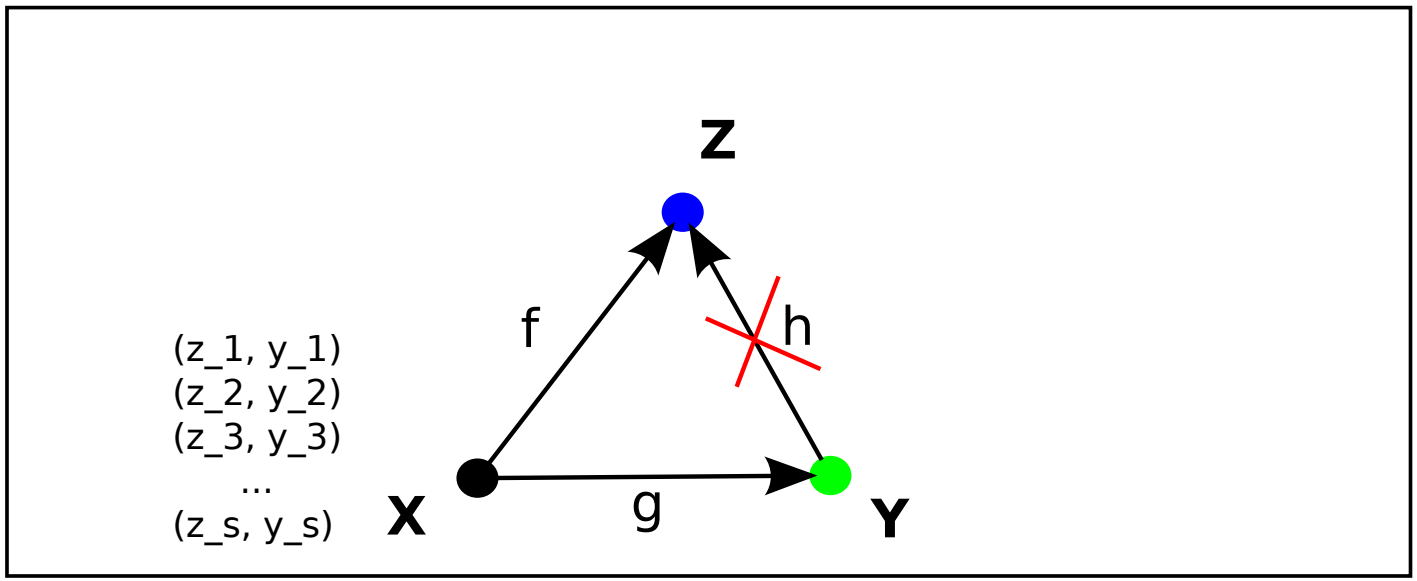


Step (4/4): Enumerating solutions.

For each blue node v in the graph, in case it would be assigned as z, we follow the f and h edges of v and obtain nodes u (unique candidate for x) and w (unique anti-candidate for y). We *register* at u pair v, w (meaning that v can work with any node except w).

Step (4/4): Enumerating solutions.

For each blue node v in the graph, in case it would be assigned as z, we follow the f and h edges of v and obtain nodes u (unique candidate for x) and w (unique anti-candidate for y). We *register* at u pair v, w (meaning that v can work with any node except w).

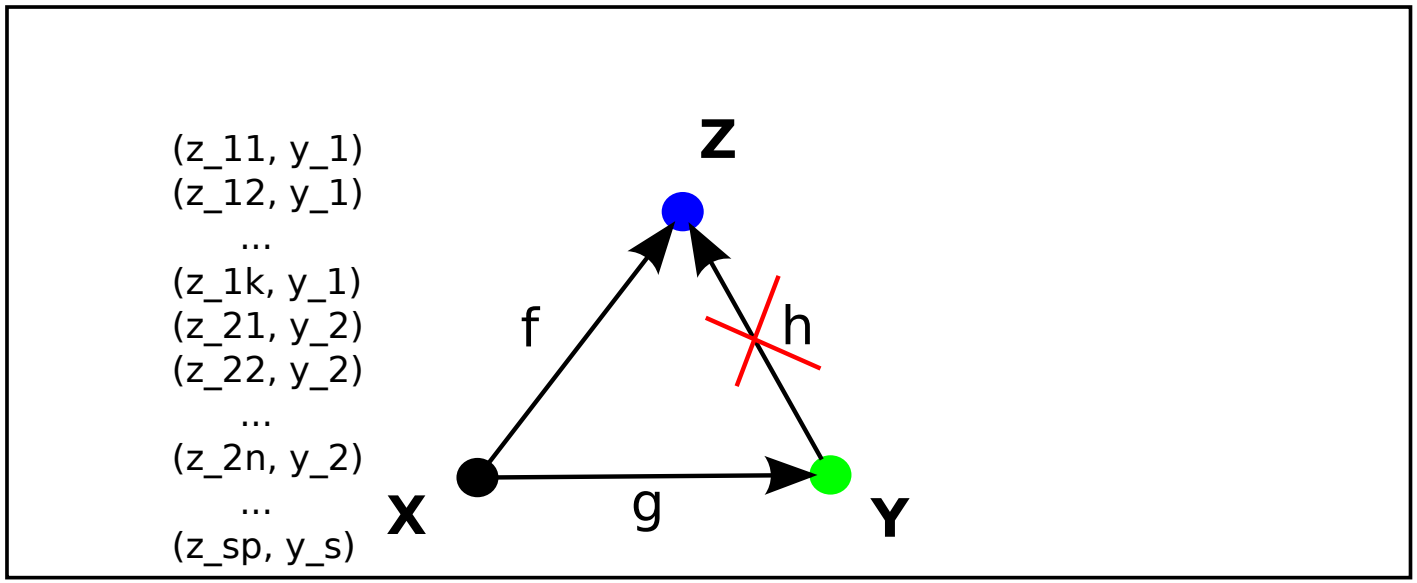


Step (4/4): Enumerating solutions.

For each black node v in the graph, in case it would be assigned as x, we sort the above lists by the y component.
Note that each such list can have arbitrary size but their sum is linear in the size of the graph (each node can register at constantly many nodes).

Step (4/4): Enumerating solutions.

For each black node v in the graph, in case it would be assigned as x, we sort the above lists by the y component.
Note that each such list can have arbitrary size but their sum is linear in the size of the graph (each node can register at constantly many nodes).

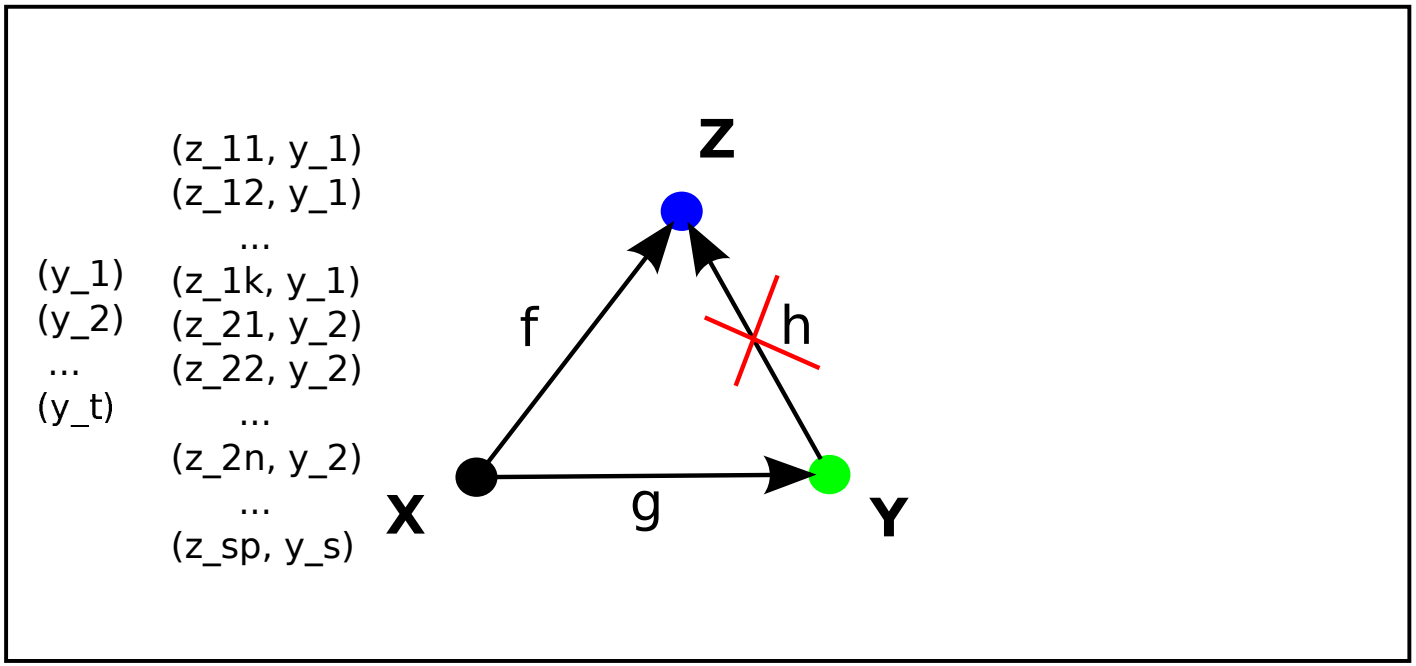


Step (4/4): Enumerating solutions.

For each green node v in the graph, in case it would be assigned as y, we follow the g edge of v and obtain node u (unique candidate for x). We *register* at u node v.

Step (4/4): Enumerating solutions.

For each green node v in the graph, in case it would be assigned as y, we follow the g edge of v and obtain node u (unique candidate for x). We *register* at u node v.



Step (4/4): Enumerating solutions.

All the above operations were done in time linear in the size of the input graph and the obtain structure is our index structure for the enumeration algorithm.

Step (4/4): Enumerating solutions.

The enumerating algorithm is then as follows:

Step (4/4): Enumerating solutions.

The enumerating algorithm is then as follows:
We go through all black nodes x.
For each such node we go through the list of y-s.
For each (x, y) we go through the list of (z_i, y_i).

Step (4/4): Enumerating solutions.

The enumerating algorithm is then as follows:
We go through all black nodes x.
For each such node we go through the list of y-s.
For each (x, y) we go through the list of (z_i, y_i).
If $y \neq y_i$ we output triple (x, y, z).
If $y = y_i$ we quickly jump to the next (z_j, y_j) such that $y_i \neq y_j$ (we can precompute this jumps upfront). As the (z, y) list was sorted by the y column, we can safely output all the remaining triples.

Thank You!

Thank You!
Any questions?