Algorithmic Meta-Theorems

192.122

WS21/22

Jan Dreier

dreier@ac.tuwien.ac.at

ALGORITHMS AND COMPLEXITY GROUP

TU WIEN Informatics

Summary

Courcelle's theorem is a very powerful tool to solve problems on bounded treewidth. It comes in various flavours.

- MSO_1: base variant,
- MSO_2: edge quantifiers,
- CMSO: parity/modulo counting,
- LinEMSOL: optimization,
- and any combination thereof.

We now want to prove the following.

Courcelle's Theorem

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time $f(\mathrm{tw}(G), |\varphi|)n$ for some function f.

Proving Courcelle's Theorem

We now want to prove the following.

Courcelle's Theorem

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time $f(\mathrm{tw}(G), |\varphi|)n$ for some function f.

- Historically proven by converting MSO_1-formulas into tree-automata. Use this automaton to traverse the tree-decomposition.

Proving Courcelle's Theorem

We now want to prove the following.

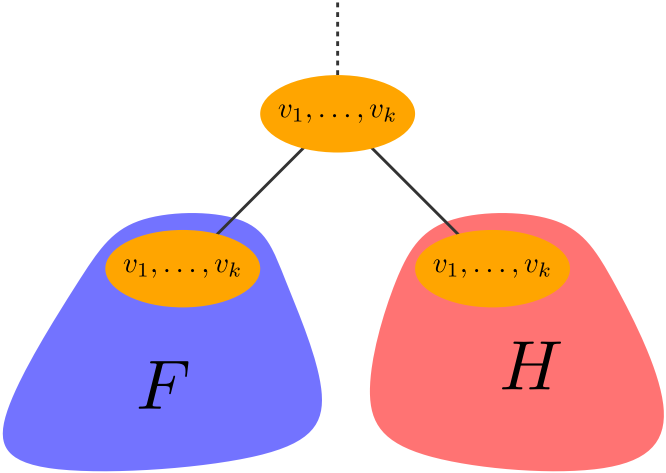
Courcelle's Theorem

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time $f(\mathrm{tw}(G), |\varphi|)n$ for some function f.

- Historically proven by converting MSO_1-formulas into tree-automata. Use this automaton to traverse the tree-decomposition.
- We prove it using a powerful logic-theorem by Fefermann and Vaught as a blackbox.

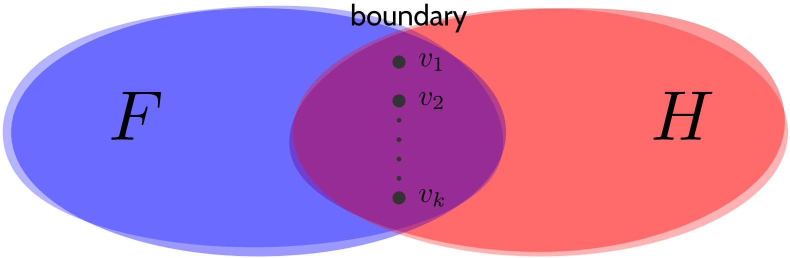
Join Nodes

We can assume we are given a nice tree decomposition. If we manage the *join* operation, *introduce* and *forget* are easy.



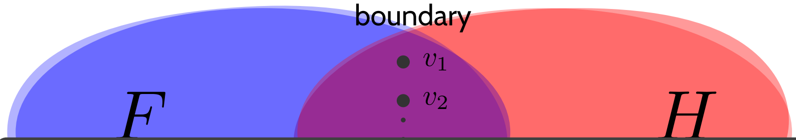
Fefermann–Vaught

Let H be a graph with boundary $v_1, \ldots, v_k$. We define q-type$(H; v_1, \ldots, v_k)$ to be the set of all MSO_1-formulas $\xi(x_1, \ldots, x_k)$ of quantifier-rank $\leq q$ with $H \models \xi(v_1, \ldots, v_k)$.



Fefermann–Vaught

Let H be a graph with boundary $v_1, \ldots, v_k$. We define q-type$(H; v_1, \ldots, v_k)$ to be the set of all MSO$_1$-formulas $\xi(x_1, \ldots, x_k)$ of quantifier-rank $\leq q$ with $H \models \xi(v_1, \ldots, v_k)$.



Theorem (Fefermann–Vaught)

Let F and H be graphs with $V(F) \cap V(H) = \{v_1, \ldots, v_k\}$. If we know

- q-type$(F; v_1, \ldots, v_k)$
- q-type$(H; v_1, \ldots, v_k)$

then we can compute q-type$(F \cup H; v_1, \ldots, v_k)$.

Dynamic Programming

We have a nice tree decomposition of a graph G and want to know whether $G \models \varphi$ for a formula with quantifier-rank q.

We have the Fefermann–Vaught theorem that tells us how to aggregate q-types when joining two subgraphs.

Dynamic Programming

We have a nice tree decomposition of a graph G and want to know whether $G \models \varphi$ for a formula with quantifier-rank q.

We have the Fefermann–Vaught theorem that tells us how to aggregate q-types when joining two subgraphs.

For bag i (with boundary $v_1, \ldots, v_k$) we store for each formula $\xi(x_1, \ldots, x_k)$ with quantifier-rank $\leq q$ a table entry

$$M_i(\xi) = \begin{cases} 1 & G[V_i] \models \xi(v_1, \ldots, v_k) \\ 0 & \text{otherwise.} \end{cases}$$

Dynamic Programming

We have a nice tree decomposition of a graph G and want to know whether $G \models \varphi$ for a formula with quantifier-rank q.

We have the Fefermann–Vaught theorem that tells us how to aggregate q-types when joining two subgraphs.

For bag i (with boundary $v_1, \ldots, v_k$) we store for each formula $\xi(x_1, \ldots, x_k)$ with quantifier-rank $\leq q$ a table entry

$$M_i(\xi) = \begin{cases} 1 & G[V_i] \models \xi(v_1, \ldots, v_k) \\ 0 & \text{otherwise.} \end{cases}$$

Let r be the root-node. Then $G \models \varphi$ iff $G[V_r] \models \varphi$ iff $M_r(\varphi) = 1$.

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\mathrm{tw}(G), |\varphi|)n$.
Dynamic programming is only fast if the tables are small.

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$.
Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G) + 1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$.
Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G) + 1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in φ.

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$.
Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G) + 1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in φ.
- "Normalize" all formulas:

$$\exists x \, x = x \wedge x = x \wedge x = x \wedge x = x \wedge x = x \wedge x = x \wedge x = x \wedge \dots$$

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$.
Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G)+1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in φ.
- "Normalize" all formulas:

$$\exists x\, x = x$$

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$. Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G) + 1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in φ.
- "Normalize" all formulas:

$$\exists x\, x = x$$

Show the claim by induction.

Size of q-types

We want to decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$. Dynamic programming is only fast if the tables are small.

We have to show that the number of formulas ξ of quantifier-rank $\leq q$ with $\leq \text{tw}(G) + 1$ free variables is bounded by some function $f(\text{tw}(G), |\varphi|)$. This bounds the number table entries ξ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in φ.
- "Normalize" all formulas:

$$\exists x\, x = x$$

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses).

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$
- $x_i {\sim} x_j$ for $1 \leq i, j \leq k$

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$
- $x_i \sim x_j$ for $1 \leq i, j \leq k$

plus their negations.

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$
- $x_i {\sim} x_j$ for $1 \leq i, j \leq k$

plus their negations.
Number of literals: $O(k^2|\varphi|)$.

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$
- $x_i \sim x_j$ for $1 \leq i, j \leq k$

plus their negations.
Number of literals: $O(k^2|\varphi|)$.
Number of clauses (after removing duplicate literals): $2^{O(k^2|\varphi|)}$

Size of q-types

Show the claim by induction. Base case $q = 0$: There are only $2^{2^{O(k^2 \cdot |\varphi|)}}$ many quantifier-free formulas with $\leq k$ variables.

We can assume the formula to be in CNF (conjunction of disjunctive clauses). The possible literals are

- $c_i(x_j)$ for $1 \leq i \leq |\varphi|$ and $1 \leq j \leq k$
- $x_i = x_j$ for $1 \leq i, j \leq k$
- $x_i \sim x_j$ for $1 \leq i, j \leq k$

plus their negations.
Number of literals: $O(k^2|\varphi|)$.
Number of clauses (after removing duplicate literals): $2^{O(k^2|\varphi|)}$
Number of formulas (after removing duplicate clauses): $2^{2^{O(k^2|\varphi|)}}$

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables.

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables. Assume we have formulas $\xi_1, \ldots, \xi_l$ with quantifier-rank $\leq q-1$ and $\leq k+1$ free variables.

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables. Assume we have formulas $\xi_1, \ldots, \xi_l$ with quantifier-rank $\leq q-1$ and $\leq k+1$ free variables.

Formulas with of quantifier-rank $\leq q$ and $\leq k$ free variables are of the form

$$\begin{aligned}
&\big(\forall x \xi_1 \wedge \exists x \xi_4 \wedge \exists x \xi_8 \wedge \ldots\big) \vee \\
&\big(\exists x \xi_3 \wedge \forall x \xi_2 \wedge \exists x \xi_9 \wedge \forall x \xi_1 \wedge \ldots\big) \vee \\
&\big(\exists x \xi_5 \wedge \forall x \xi_8 \wedge \ldots\big) \vee \ldots
\end{aligned}$$

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables. Assume we have formulas $\xi_1, \ldots, \xi_l$ with quantifier-rank $\leq q-1$ and $\leq k+1$ free variables.

Formulas with of quantifier-rank $\leq q$ and $\leq k$ free variables are of the form

$$\begin{aligned}
&\big(\forall x \xi_1 \wedge \exists x \xi_4 \wedge \exists x \xi_8 \wedge \ldots\big) \vee \\
&\big(\exists x \xi_3 \wedge \forall x \xi_2 \wedge \exists x \xi_9 \wedge \forall x \xi_1 \wedge \ldots\big) \vee \\
&\big(\exists x \xi_5 \wedge \forall x \xi_8 \wedge \ldots\big) \vee \ldots
\end{aligned}$$

There are at most $2^{2^{2l}}$ of them.

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables. Assume we have formulas $\xi_1, \ldots, \xi_l$ with quantifier-rank $\leq q-1$ and $\leq k+1$ free variables.

Formulas with of quantifier-rank $\leq q$ and $\leq k$ free variables are of the form

$$\begin{aligned}&\big(\forall x \xi_1 \wedge \exists x \xi_4 \wedge \exists x \xi_8 \wedge \ldots\big) \vee \\ &\big(\exists x \xi_3 \wedge \forall x \xi_2 \wedge \exists x \xi_9 \wedge \forall x \xi_1 \wedge \ldots\big) \vee \\ &\big(\exists x \xi_5 \wedge \forall x \xi_8 \wedge \ldots\big) \vee \ldots\end{aligned}$$

There are at most $2^{2^{2l}}$ of them. In total, the number of formulas is roughly

$$\underbrace{2^{\cdot^{\cdot^{2^{O(\mathsf{tw}(G)^2 \cdot |\varphi|)}}}}}_{2q}.$$

Size of q-types

Next, we bound number of formulas with quantifier-rank $\leq q$ and $\leq k$ free variables. Assume we have formulas $\xi_1, \dots, \xi_l$ with quantifier-rank $\leq q-1$ and $\leq k+1$ free variables.

Formulas with of quantifier-rank $\leq q$ and $\leq k$ free variables are of the form

$$\begin{aligned}&\big(\forall x\xi_1 \wedge \exists x\xi_4 \wedge \exists x\xi_8 \wedge \dots\big)\vee\\&\big(\exists x\xi_3 \wedge \forall x\xi_2 \wedge \exists x\xi_9 \wedge \forall x\xi_1 \wedge \dots\big)\vee\\&\big(\exists x\xi_5 \wedge \forall x\xi_8 \wedge \dots\big) \vee \dots\end{aligned}$$

There are at most $2^{2^{2l}}$ of them. In total, the number of formulas is roughly

$$\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{O(\mathsf{tw}(G)^2 \cdot |\varphi|)}}}}}}_{2q}.$$

This bound cannot be improved much.

Dynamic Programming

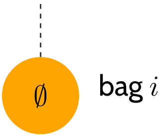
Remember: For bag i (with boundary $v_1, \ldots, v_k$) we store for each formula $\xi(x_1, \ldots, x_k)$ with quantifier-rank $\leq q$ a table entry

$$M_i(\xi) = \begin{cases} 1 & G[V_i] \models \xi(v_1, \ldots, v_k) \\ 0 & \text{otherwise.} \end{cases}$$

We now compute these values bottom-up.

Leaf Nodes

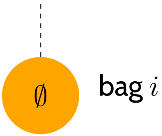
$G[V_i]$ is the empty graph.



Leaf Nodes

$G[V_i]$ is the empty graph.

$M_i(\xi)$: All sentences of quantifier-rank $\leq q$ that hold in the empty graph.

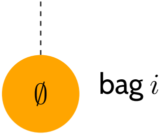


Leaf Nodes

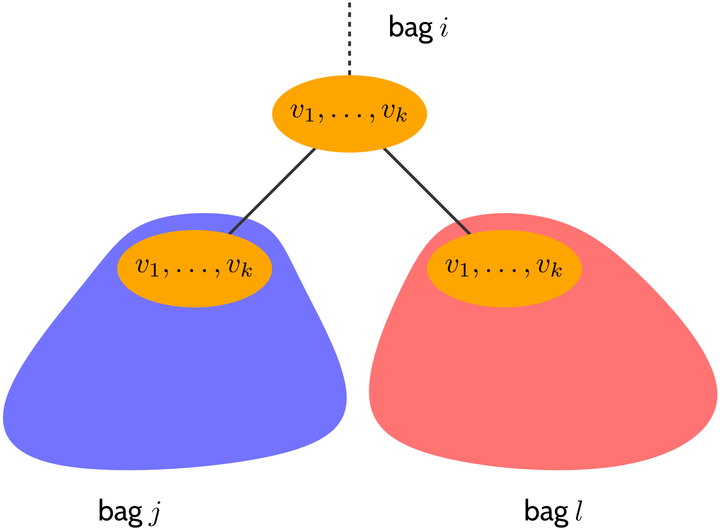
$G[V_i]$ is the empty graph.

$M_i(\xi)$: All sentences of quantifier-rank $\leq q$ that hold in the empty graph.

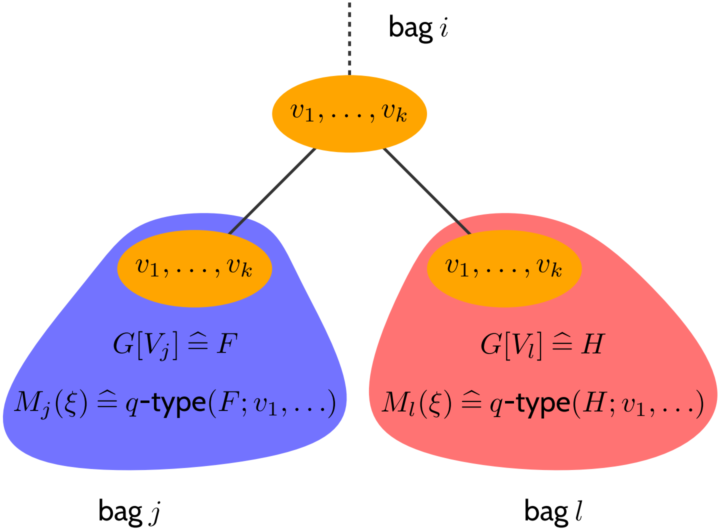
Simply evaluate them.



Join Nodes

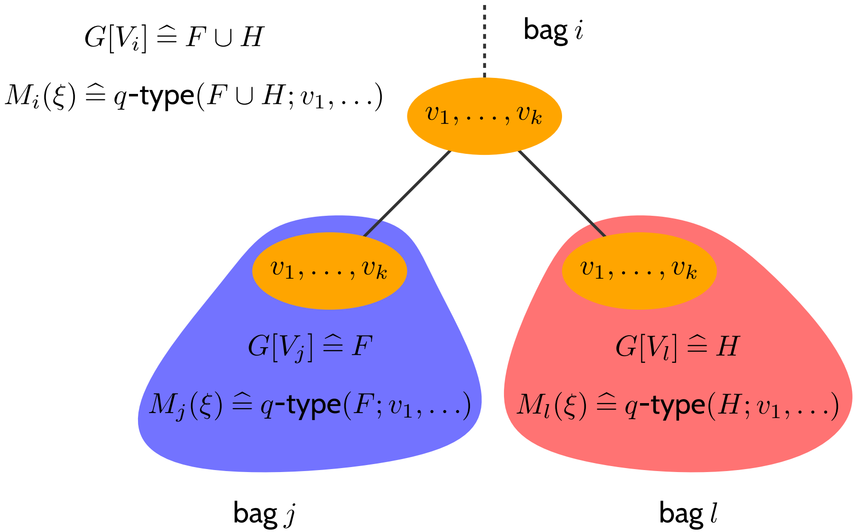


Join Nodes



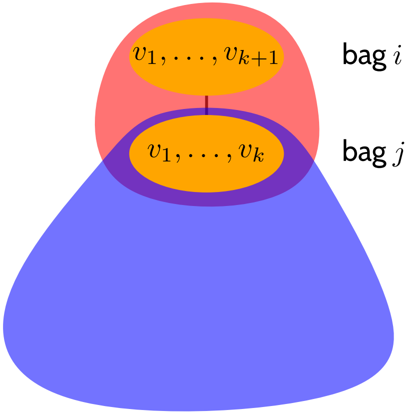
bag i
$v_1, \ldots, v_k$
$v_1, \ldots, v_k$
$v_1, \ldots, v_k$
$G[V_j] \mathrel{\widehat{=}} F$
$M_j(\xi) \mathrel{\widehat{=}} q\text{-type}(F; v_1, \ldots)$
$G[V_l] \mathrel{\widehat{=}} H$
$M_l(\xi) \mathrel{\widehat{=}} q\text{-type}(H; v_1, \ldots)$
bag j
bag l

Join Nodes

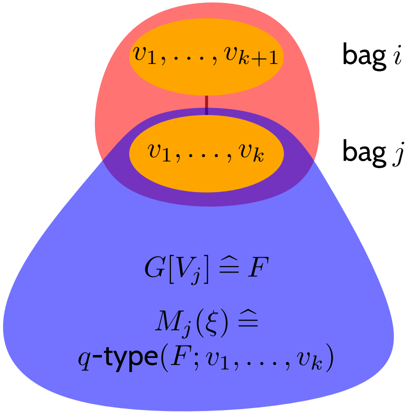


$G[V_i] \mathrel{\widehat{=}} F \cup H$
$M_i(\xi) \mathrel{\widehat{=}} q\text{-type}(F \cup H; v_1, \ldots)$
bag i
$v_1, \ldots, v_k$
$v_1, \ldots, v_k$
$G[V_j] \mathrel{\widehat{=}} F$
$M_j(\xi) \mathrel{\widehat{=}} q\text{-type}(F; v_1, \ldots)$
$v_1, \ldots, v_k$
$G[V_l] \mathrel{\widehat{=}} H$
$M_l(\xi) \mathrel{\widehat{=}} q\text{-type}(H; v_1, \ldots)$
bag j
bag l

Introduce Nodes



Introduce Nodes

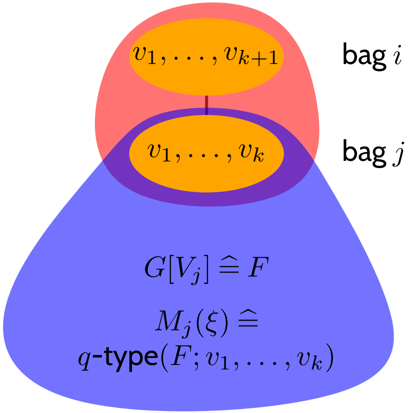


Introduce Nodes

$$G[\{v_1, \ldots, v_{k+1}\}] \mathrel{\widehat{=}} H$$

compute

q-type$(H; v_1, \ldots, v_{k+1})$

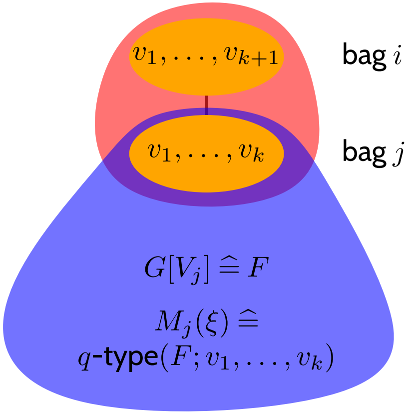


Introduce Nodes

$$G[\{v_1, \dots, v_{k+1}\}] \mathrel{\widehat{=}} H$$

$$G[V_i] \mathrel{\widehat{=}} F \cup H$$
$$M_i(\xi) \mathrel{\widehat{=}} q\text{-type}(F \cup H; v_1, \dots, v_{k+1})$$

compute
q-type$(H; v_1, \dots, v_{k+1})$



Introduce Nodes

$G[\{v_1, \ldots, v_{k+1}\}] \mathrel{\widehat{=}} H$

$G[V_i] \mathrel{\widehat{=}} F \cup H$

$M_i(\xi) \mathrel{\widehat{=}} q\text{-type}(F \cup H; v_1, \ldots, v_{k+1})$

compute

$q\text{-type}(H; v_1, \ldots, v_{k+1})$

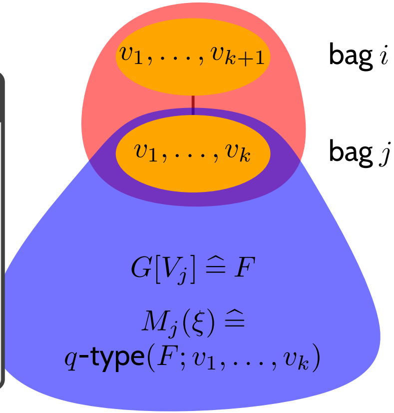
Theorem (Fefermann–Vaught)

If we know

- $q\text{-type}(F; v_1, \ldots, v_k)$
- $q\text{-type}(H; v_1, \ldots, v_k \qquad)$

then we can compute

$q\text{-type}(F \cup H; v_1, \ldots, v_k \qquad)$.

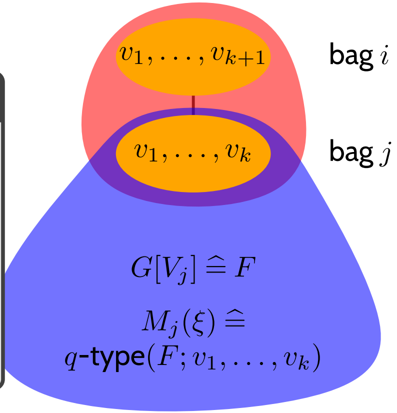


Introduce Nodes

$G[\{v_1, \ldots, v_{k+1}\}] \mathrel{\widehat{=}} H$

$G[V_i] \mathrel{\widehat{=}} F \cup H$
$M_i(\xi) \mathrel{\widehat{=}} q\text{-type}(F \cup H; v_1, \ldots, v_{k+1})$

compute
$q\text{-type}(H; v_1, \ldots, v_{k+1})$



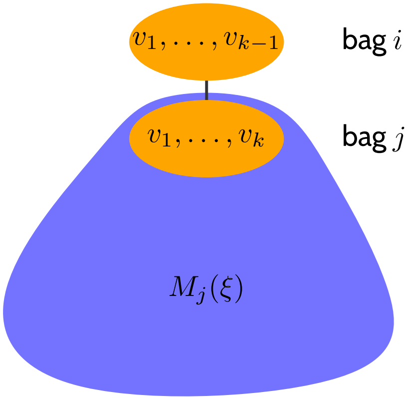
Theorem (Fefermann–Vaught)

If we know

- $q\text{-type}(F; v_1, \ldots, v_k)$
- $q\text{-type}(H; v_1, \ldots, v_k, v_{k+1})$

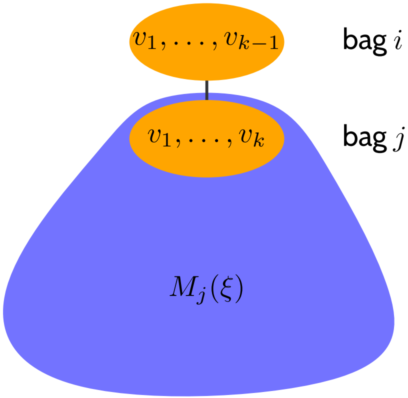
then we can compute
$q\text{-type}(F \cup H; v_1, \ldots, v_k, v_{k+1})$.

Forget Nodes



Forget Nodes

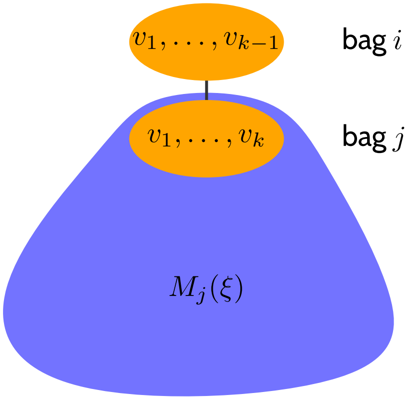
$$G[V_i] = G[V_j]$$



Forget Nodes

$G[V_i] = G[V_j]$

$M_j(\xi)$: Formulas over $x_1, \ldots, x_k$.

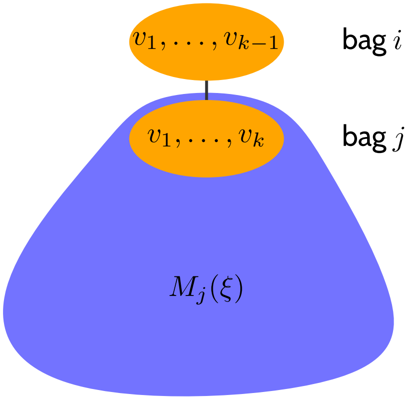


Forget Nodes

$G[V_i] = G[V_j]$

$M_j(\xi)$: Formulas over $x_1, \ldots, x_k$.

$M_i(\xi)$: Formulas over $x_1, \ldots, x_{k-1}$.



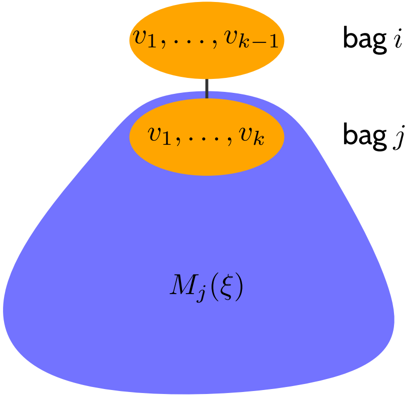
Forget Nodes

$G[V_i] = G[V_j]$

$M_j(\xi)$: Formulas over $x_1, \ldots, x_k$.

$M_i(\xi)$: Formulas over $x_1, \ldots, x_{k-1}$.

Remove all formulas that mention x_k.



Final Result

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time

$$\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{O(\mathsf{tw}(G)^2|\varphi|)}}}}}}_{2|\varphi|} n.$$

Final Result

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time

$$\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{O(\mathrm{tw}(G)^2|\varphi|)}}}}}}_{2|\varphi|} n.$$

Can this be improved?

Final Result

For a MSO_1 formula φ and graph G one can decide whether $G \models \varphi$ in time

$$\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{O(\mathsf{tw}(G)^2|\varphi|)}}}}}}_{2|\varphi|} n.$$

Can this be improved?

No. It cannot be done without such a tower of powers (Frick, Grohe 2004).

Implementations

Usually, Courcelle’s theorem is considered a theoretical classification tool.

Implementations

Usually, Courcelle's theorem is considered a theoretical classification tool. But it has been implemented by Kneis, Langer, and Rossmanith.

Implementations

Usually, Courcelle's theorem is considered a theoretical classification tool. But it has been implemented by Kneis, Langer, and Rossmanith.

- Doing it naively has horrible, horrible run time …

Implementations

Usually, Courcelle's theorem is considered a theoretical classification tool. But it has been implemented by Kneis, Langer, and Rossmanith.

- Doing it naively has horrible, horrible run time ...
- By storing "game trees" instead, it becomes feasible `https://github.com/sequoia-mso`.

How to formulate Problems for Sequoia

```
-- More efficient formula for three-coloring; tests whether
-- (R, G, V\setminus (R\cup G)) is a proper three-coloring of the graph

ThreeCol(R, B) :=
        All x (
                (x notin R or x notin B)
            and
                All y (
                    ~adj(x,y) or (
                        (x notin R or y notin R) and
                        (x notin B or y notin B) and
                        ((x in R) or (x in B)
                                                      or
                         (y in R) or (y in B))
                    )
                )
        )
```

Proof for Extensions

- We saw the proof for the base variant MSO_1.

Proof for Extensions

- We saw the proof for the base variant MSO_1.
- By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant CMSO.

Proof for Extensions

- We saw the proof for the base variant MSO$_1$.
- By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant CMSO.
- Also we already know how to reduce MSO$_2$ to MSO$_1$.

Proof for Extensions

- We saw the proof for the base variant MSO_1.
- By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant CMSO.
- Also we already know how to reduce MSO_2 to MSO_1.
- The proof for the optimization variant LinEMSOL works similarly by also keeping track of the largest satisfying assignment.

How about Dense Graphs?

- A graph containing a clique of size k has treewidth at least $k-1$.

How about Dense Graphs?

- A graph containing a clique of size k has treewidth at least $k-1$.
- There is a width measure *cliquewidth* similar to treewidth for which cliques have width one.

How about Dense Graphs?

- A graph containing a clique of size k has treewidth at least $k - 1$.
- There is a width measure *cliquewidth* similar to treewidth for which cliques have width one.
- Courcelle's theorem for MSO_1 also holds for cliquewidth.

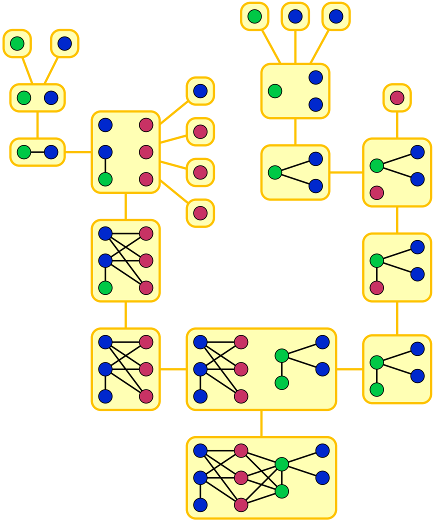
How about Dense Graphs?

- A graph containing a clique of size k has treewidth at least $k - 1$.
- There is a width measure *cliquewidth* similar to treewidth for which cliques have width one.
- Courcelle's theorem for MSO_1 also holds for cliquewidth.
- On the other hand MSO_2 only holds for treewidth.

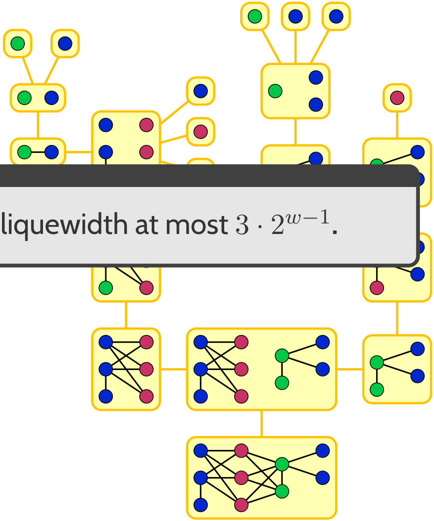
Cliquewidth

Cliquewidth cw(G): Minimum number of colors needed to construct G using these operations.

- Creation of new vertex with color i
- Disjoint union of two graphs
- Joining by an edge every vertex with color i to every vertex with color j
- Changing color i to color j



Cliquewidth cw(G): Minimum number of colors needed to construct G using these operations.

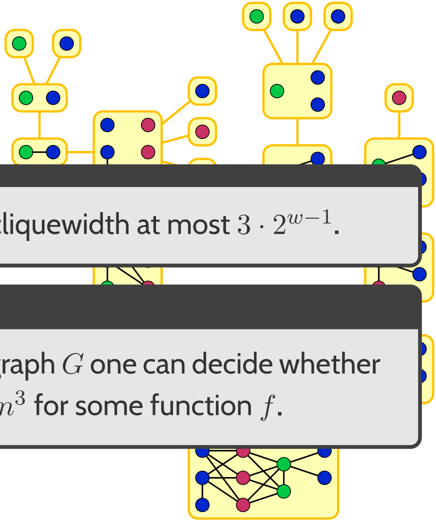


Graphs of treewidth w have cliquewidth at most $3 \cdot 2^{w-1}$.

- Disjoint union of two graphs
- Joining by an edge every vertex with color i to every vertex with color j
- Changing color i to color j

Cliquewidth

Cliquewidth cw(G): Minimum number of colors needed to construct G using these operations.



Graphs of treewidth w have cliquewidth at most $3 \cdot 2^{w-1}$.

Courcelle's Theorem

For a MSO_1 sentence φ and graph G one can decide whether $G \models \varphi$ in time $f(\text{cw}(G), |\varphi|)n^3$ for some function f.

- vertex with color j
- Changing color i to color j

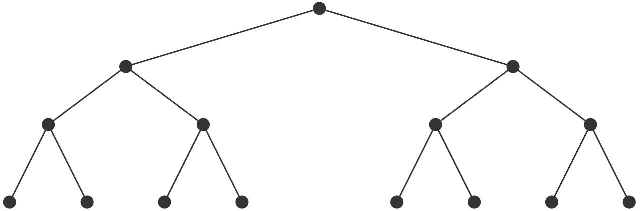
And Now For Something Completely Different...

Let us go back to the first lecture.

Independent Set on Trees

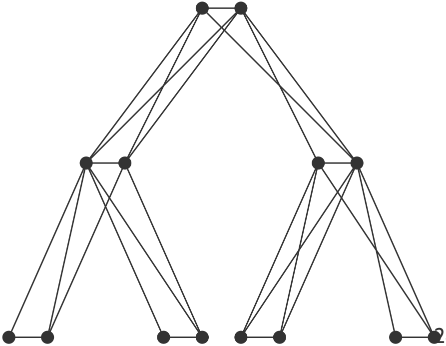
INDEPENDENTSET can be solved in linear time on trees.

Idea: Root the tree and do dynamic programming. Starting at the leafs, compute for each subtree the maximum size of a solution with and without its root.



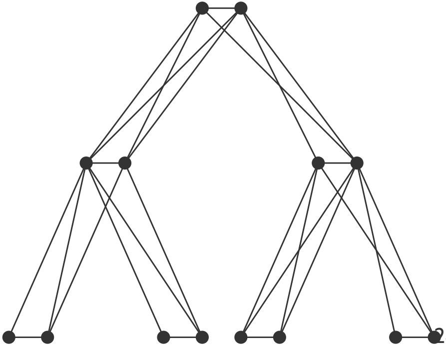
First Main Result

- This approach can be extended to tree-like graphs (bounded treewidth).



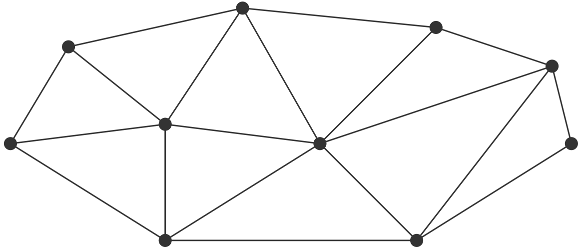
First Main Result

- This approach can be extended to tree-like graphs (bounded treewidth).
- First main result of the lecture (Courcelle's theorem): Every problem definable in monadic second-order logic can be solved in linear time on graphs of bounded treewidth.
- This includes
  - coloring
  - independent set
  - clique
  - dominating set
  - feedback vertex set
  - hamilton path
  - ...



Independent Set on Planar Graphs

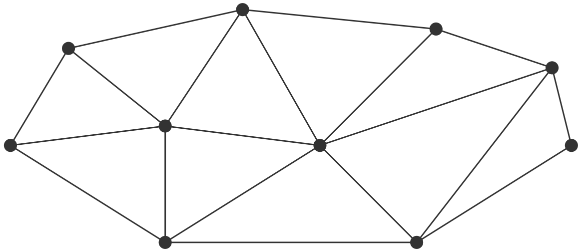
How about planar graphs?



Independent Set on Planar Graphs

How about planar graphs?

INDEPENDENTSET is NP-complete on planar graphs.



Algorithm

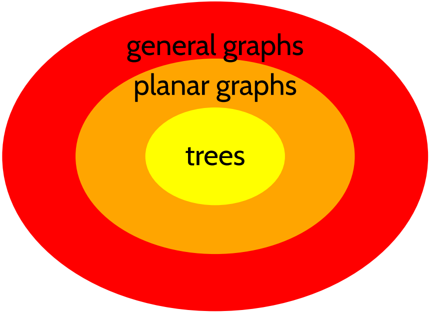
One can decide whether a planar graph has an independent set of size k in time $O(6^k n)$.

IS(G, k):
    if G is empty return $k == 0$
    find vertex v with degree ≤ 5 in G
    for all $w \in N(v)$:
        if IS($G \setminus N(w)$, $k - 1$) return True
    return False

Summary

INDEPENDENTSET is hard on general graphs. However,

- on trees, we can solve it in linear time
- on planar graphs, it is still fixed parameter tractable.

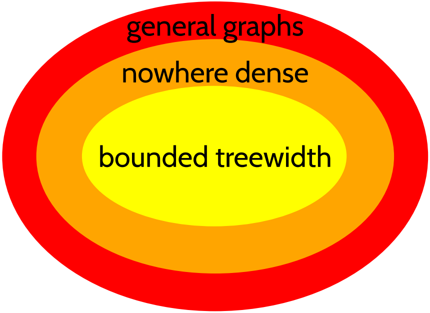


We will observe a similar behaviour for many other problems!

Summary

INDEPENDENTSET is hard on general graphs. However,

- on bounded treewidth, we can solve it in linear time
- on nowhere dense graphs, it is still fixed parameter tractable.



We will observe a similar behaviour for many other problems!

Graph Classes

- *Width measures* (treewidth, degree, . . .) capture the structure of a graph using *one number*. Sometimes, we may need more numbers to describe something.

Graph Classes

- *Width measures* (treewidth, degree, ...) capture the structure of a graph using *one number*. Sometimes, we may need more numbers to describe something.
- From now on, we work with (infinite) *graph classes*.

Graph Classes

- A class $\mathcal{C}$ has *bounded treewidth* if there exists a constant c such that for all $G \in \mathcal{C}$ holds $\text{tw}(G) \leq c$.

Graph Classes

- A class $\mathcal{C}$ has *bounded treewidth* if there exists a constant c such that for all $G \in \mathcal{C}$ holds $\text{tw}(G) \leq c$.
- Attention! *Bounded treewidth* is a property of graph *classes* not of graphs!

Graph Classes

- A class $\mathcal{C}$ has *bounded treewidth* if there exists a constant c such that for all $G \in \mathcal{C}$ holds $\text{tw}(G) \leq c$.
- Attention! *Bounded treewidth* is a property of graph *classes* not of graphs!
- Generally, a class has bounded X if there is a constant c such that for all $G \in \mathcal{C}$ holds $X \leq c$.

Graph Classes

- A class $\mathcal{C}$ has *bounded treewidth* if there exists a constant c such that for all $G \in \mathcal{C}$ holds $\text{tw}(G) \leq c$.
- Attention! *Bounded treewidth* is a property of graph *classes* not of graphs!
- Generally, a class has bounded X if there is a constant c such that for all $G \in \mathcal{C}$ holds $X \leq c$.
- Assume we have a bounded treewidth class $\mathcal{C}$. On this class, Coucelle's theorem solves MSO_1 formulas in time $f(|\varphi|, \text{tw}(G))n \leq f(|\varphi|, c)n = f'(|\varphi|)n$.

Restating Courcelle

Courcelle's Theorem

Let $\mathcal{C}$ be a graph class with bounded treewidth. There exsists a function f (depending on $\mathcal{C}$!) such that for every MSO$_1$ sentence φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

Restating Courcelle

Courcelle's Theorem

Let $\mathcal{C}$ be a graph class with bounded treewidth. There exsists a function f (depending on $\mathcal{C}$!) such that for every MSO_1 sentence φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$.

If a fixed problem is expressible by some formula φ, then $f(|\varphi|) = O(1)$.

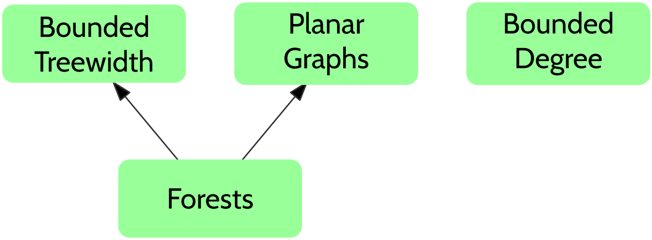
Courcelle's Theorem (most succinct formulation)

On graph classes with bounded treewidth, one can decide MSO_1-expressible problems in linear time.

Inclusion Diagrams

Each box represents a property of graph classes.

What do the arrows mean?



Sparsity

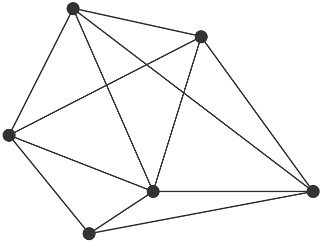
- What do these graphs have in common?
  - Graphs with treewidth w have at most wn edges.
  - Planar graphs have at most $3n$ edges.
  - Graphs with constant degree have $O(n)$ edges.

Sparsity

- What do these graphs have in common?
  - Graphs with treewidth w have at most wn edges.
  - Planar graphs have at most $3n$ edges.
  - Graphs with constant degree have $O(n)$ edges.
- Problems seem to be easier if the graphs are *sparse*!
- What does it really mean to be sparse?

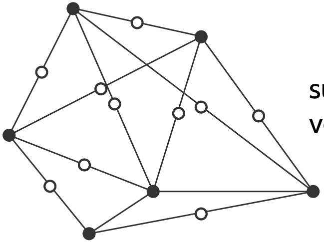
Sparsity

- Every graph is "sparse" if you subdivide the edges.



Sparsity

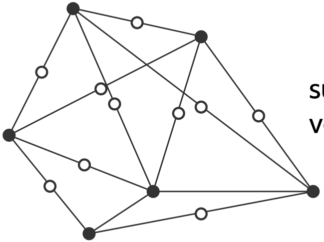
- Every graph is “sparse” if you subdivide the edges.



subdivision adds $1/2$
vertex per edge

Sparsity

- Every graph is "sparse" if you subdivide the edges.

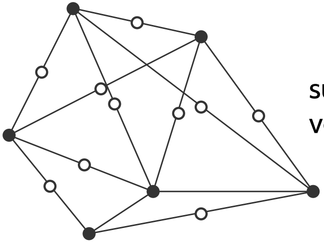


subdivision adds $1/2$
vertex per edge

- Do we consider such subdivisions sparse?
  - Yes: Degeneracy
  - No: Bounded expansion and nowhere dense graph classes

Sparsity

- Every graph is “sparse” if you subdivide the edges.



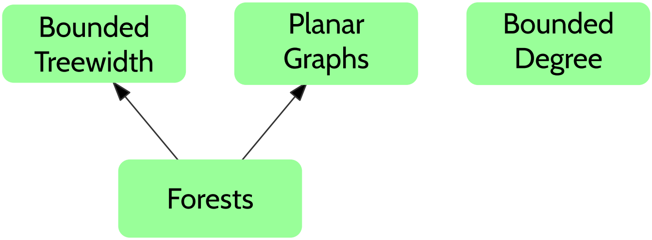
subdivision adds $1/2$
vertex per edge

- Do we consider such subdivisions sparse?
  - Yes: Degeneracy
  - No: Bounded expansion and nowhere dense graph classes
- We say “No” because it has nicer algorithmic theory.

Inclusion Diagrams

Each box represents a property of graph classes.

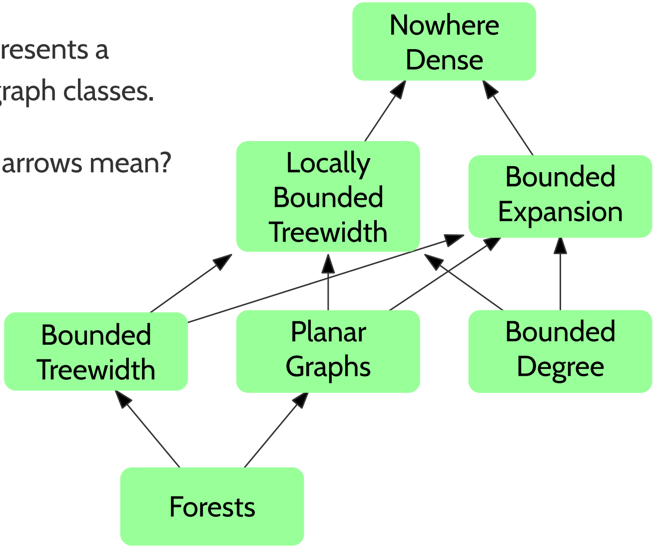
What do the arrows mean?



Inclusion Diagrams

Each box represents a property of graph classes.

What do the arrows mean?



Many Sparse Graph Classes

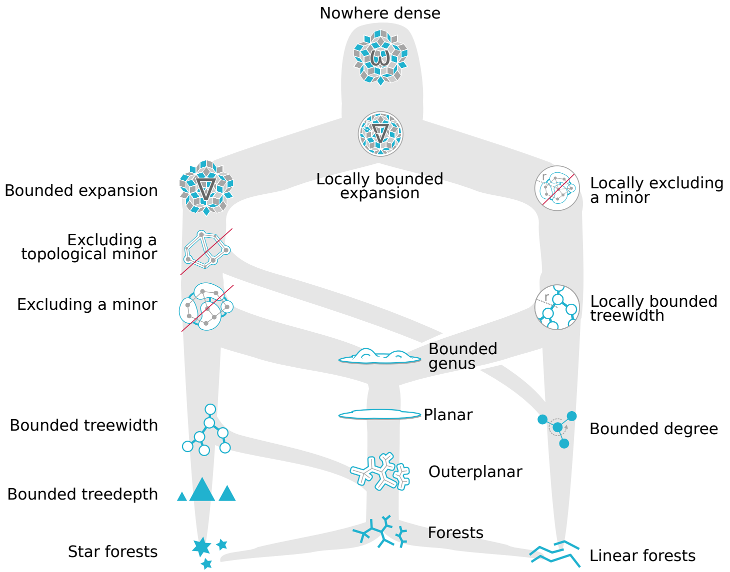


Figure by Felix Reidl

First-Order Logic

For sparse graphs, MSO_1 is too powerful. For example Independent Set, Coloring, Dominating Set are NP-complete on planar graphs or bounded degree graph classes. However, *first-order logic* fits just right.

First-Order Logic

For sparse graphs, MSO$_1$ is too powerful. For example Independent Set, Coloring, Dominating Set are NP-complete on planar graphs or bounded degree graph classes. However, *first-order logic* fits just right.

Main Result (roughly)

Let $\mathcal{C}$ be a sparse graph class. For an FO formula φ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$ for some function f.

First-Order Logic

For a given signature τ, first-order logic has ...

- element-variables $(x, y, z, \dots)$
- the equality relation $=$ as well as the relations from τ.
- quantifiers $\exists$ and $\forall$, as well as operators $\wedge$, $\vee$ and $\neg$

We mostly work on colored undirected graphs with $\tau = \{\sim, c_1, c_2, \dots\}$. Here, we call the logic FO.

Expressiveness

Can these properties be expressed in FO logic?

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

- The number of vertices is even.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \, \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

- The number of vertices is even. No.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \dots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

- The number of vertices is even. No.
- The graph is connected.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \, \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

- The number of vertices is even. No.
- The graph is connected. No.

Connection to Database Queries

Database languages such as SQL build upon first-order logic.

Connection to Database Queries

Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query $\Leftrightarrow$
  Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.

Connection to Database Queries

Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query $\Leftrightarrow$
  Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.
- Boolean query $\Leftrightarrow$ Decide whether $G \models \varphi$.

Connection to Database Queries

Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query $\Leftrightarrow$
  Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.
- Boolean query $\Leftrightarrow$ Decide whether $G \models \varphi$.
- There exist extensions of first-order logic simulating SQL's COUNT operator.

Central Problems

First-Order Model-Checking (Query Evaluation)

Input: Graph G and first-order sentence φ

Question: $G \models \varphi$?

Central Problems

First-Order Model-Checking (Query Evaluation)

Input: Graph G and first-order sentence φ

Question: $G \models \varphi$?

First-Order Query Enumeration

Input: Graph G and first-order formula $\varphi(x_1, \dots, x_k)$

Question: Enumerate all $v_1, \dots, v_k$ with $G \models \varphi(v_1, \dots, v_k)$.

Central Problems

First-Order Model-Checking (Query Evaluation)

Input: Graph G and first-order sentence φ

Question: $G \models \varphi$?

First-Order Query Enumeration

Input: Graph G and first-order formula $\varphi(x_1, \ldots, x_k)$

Question: Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.

First-Order Query Counting

Input: Graph G and first-order formula $\varphi(x_1, \ldots, x_k)$

Question: Count number of tuples $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

FO model-checking on planar graphs in NP-complete.

Proof: Reduction from Independent Set.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

Complexity

Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

FO model-checking on planar graphs in NP-complete.

Proof: Reduction from Independent Set.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

However, usually database queries are very small compared to the size of the database. Parameterize by $|\varphi|$.

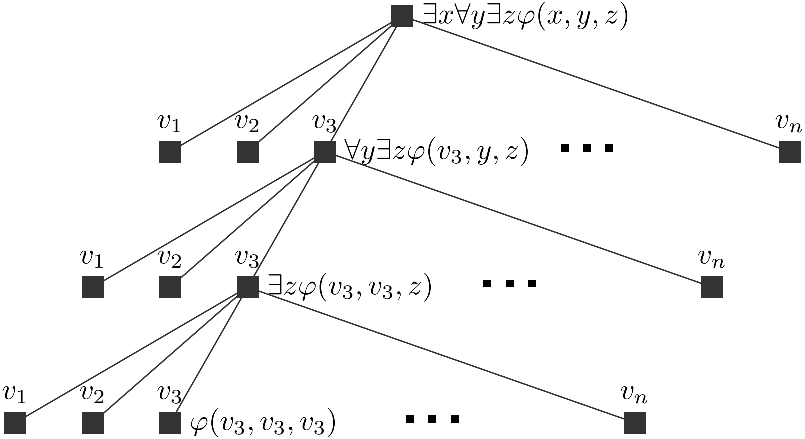
Parameterized Complexity (Upper Bound)

Theorem

One can decide whether $G \models \varphi$ in time $O(|G|^{|\varphi|})$.

Evaluation Trees

Proof: We can assume φ to be in prenex normal form. Construct an evaluation tree of size $O(|G|^{|\varphi|})$.



Parameterized Complexity (Lower Bound)

Conjecture (based on SETH)

It is believed one cannot decide whether $G \models \varphi$ in time $O(|G|^{q-1-\varepsilon})$ for any $\varepsilon > 0$ where q is the number of quantifiers of φ.

The previous algorithm is probably more or less optimal.

A faster model-checking algorithm would lead to a faster algorithm for many other problems.

On certain graph classes, we can do much better though.

Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size k.

$$\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \wedge \neg x_i = x_j$$

- There exists a dominating set of size k.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_{i} y \sim x_i \vee y = x_i$$

- The number of vertices is even. No.
- The graph is connected. No.

Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

- The *quantifier-rank* of a formula is the maximum number of nested quantifiers.

Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

- The *quantifier-rank* of a formula is the maximum number of nested quantifiers.
- We write $G \equiv_q H$ if for all first-order sentences φ of quantifier-rank $\leq q$ holds $G \models \varphi \iff H \models \varphi$.

Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

- The *quantifier-rank* of a formula is the maximum number of nested quantifiers.
- We write $G \equiv_q H$ if for all first-order sentences φ of quantifier-rank $\leq q$ holds $G \models \varphi \iff H \models \varphi$.
- Show that for every q there is a connected graph G_q and a disconnected graph H_q with $G_q \equiv_q H_q$.

Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

- The *quantifier-rank* of a formula is the maximum number of nested quantifiers.
- We write $G \equiv_q H$ if for all first-order sentences φ of quantifier-rank $\leq q$ holds $G \models \varphi \iff H \models \varphi$.
- Show that for every q there is a connected graph G_q and a disconnected graph H_q with $G_q \equiv_q H_q$.
- If there was a formula to decide connectivity it would have quantifier-rank q for some q. But this formula cannot tell G_q and H_q apart. A contradiction.

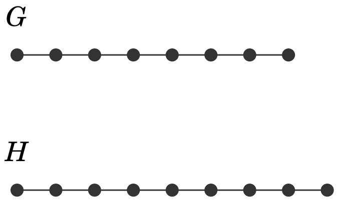
Ehrenfeucht–Fraïssé Games

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula φ such that $G \models \varphi$ iff G is connected. How do we prove that?

- The *quantifier-rank* of a formula is the maximum number of nested quantifiers.
- We write $G \equiv_q H$ if for all first-order sentences φ of quantifier-rank $\leq q$ holds $G \models \varphi \iff H \models \varphi$.
- Show that for every q there is a connected graph G_q and a disconnected graph H_q with $G_q \equiv_q H_q$.
- If there was a formula to decide connectivity it would have quantifier-rank q for some q. But this formula cannot tell G_q and H_q apart. A contradiction.
- Show $G \equiv_q H$ using *Ehrenfeucht–Fraïssé games*.

Ehrenfeucht–Fraïssé Games

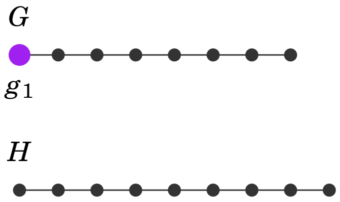
The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

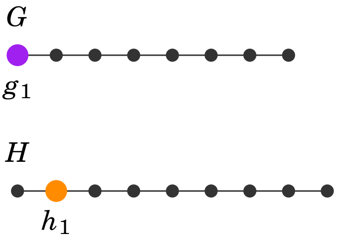
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

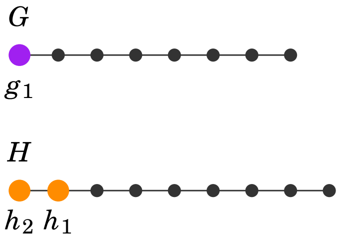
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

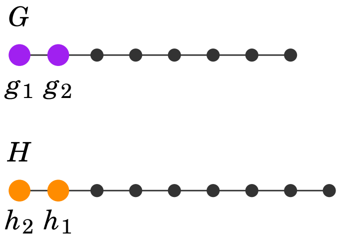
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

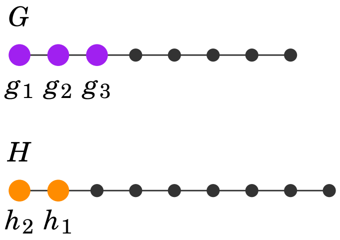
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

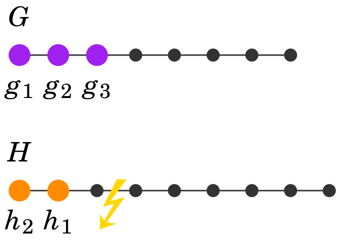
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i {\sim} g_j \iff h_i {\sim} h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

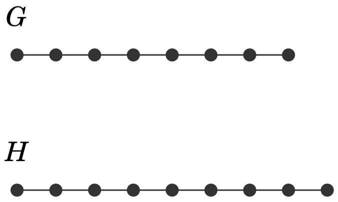
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

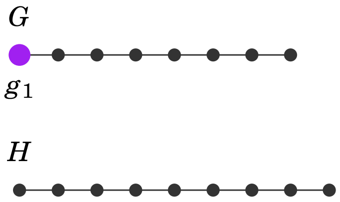
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

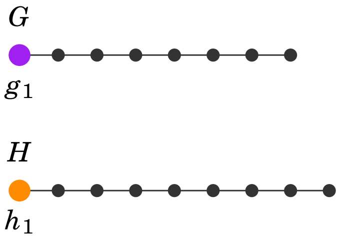
The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

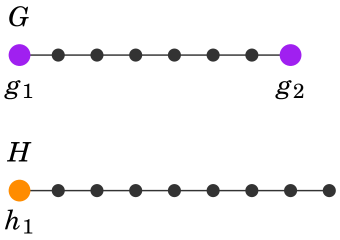
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i{\sim}g_j \iff h_i{\sim}h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

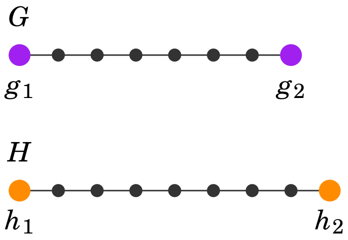
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

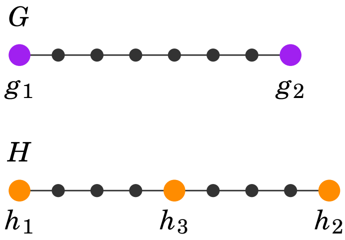
The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

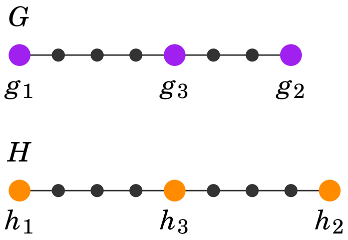
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

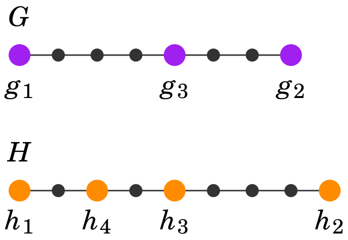
The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

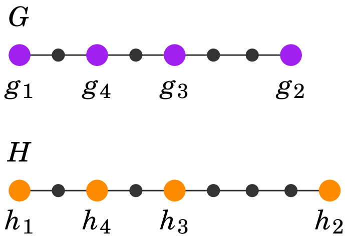
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

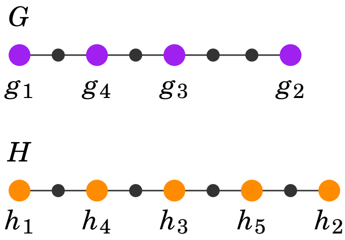
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

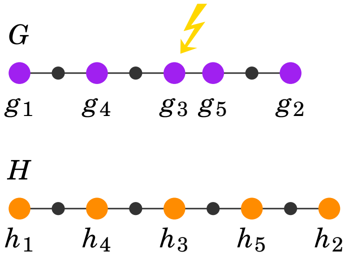
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

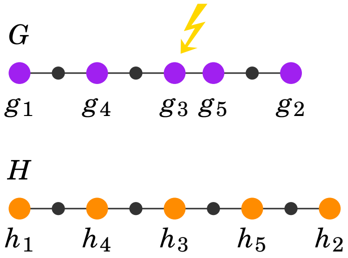
- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i \sim g_j \iff h_i \sim h_j$ for all i, j.



Ehrenfeucht–Fraïssé Games

The q-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs G and H.

- Spoiler picks $g_i \in V(G)$ or $h_i \in H(G)$
- Duplicator picks partner vertex in other graph.
- Repeat q times to get $g_1, \ldots, g_q \in V(G)$ and $h_1, \ldots, h_q \in V(H)$ (pairwise distinct).
- Duplicator wins if $g_i{\sim}g_j \iff h_i{\sim}h_j$ for all i, j.



Theorem

$G \equiv_q H$ iff the Duplicator wins the q-round Ehrenfeucht–Fraïssé game.