Algorithmic Meta-Theorems

192.122
WS21/22
Jan Dreier
dreier@ac.tuwien.ac.at
Courcelle's theorem is a very powerful tool to solve problems on bounded treewidth. It comes in various flavours.

- $\text{MSO}_1$: base variant,
- $\text{MSO}_2$: 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 $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time $f(tw(G), |\varphi|)n$ for some function $f$. 
Proving Courcelle's Theorem

We now want to prove the following.

**Courcelle's Theorem**

For a $\text{MSO}_1$ formula $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time $f(\text{tw}(G), |\varphi|)n$ for some function $f$.

- Historically proven by converting $\text{MSO}_1$-formulas into tree-automata. Use this automaton to traverse the tree-decomposition.
We now want to prove the following.

**Courcelle’s Theorem**

For a MSO$_1$ formula $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time $f(tw(G), |\varphi|)n$ 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 \textit{join} operation, \textit{introduce} and \textit{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 $H \models \xi(v_1, \ldots, v_k)$. 
Let $H$ be a graph with boundary $v_1, \ldots, v_k$. We define $q\text{-type}(H; v_1, \ldots, v_k)$ 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

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

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

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 $M_r(\varphi) = 1$. 
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 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 \( \xi \) 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 \( \xi \) in \( M_i(\xi) \).
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 $\xi$ 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 $\xi$ in $M_i(\xi)$.

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

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 $\xi$ 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 $\xi$ in $M_i(\xi)$.

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

  $$\exists x \ x = x \land x = x \land x = x \land x = x \land x = x \land x = x \land x = x \land \ldots$$
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 $\xi$ 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 $\xi$ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in $\varphi$.
- “Normalize” all formulas:
  $$\exists x \ x = x$$
We want to decide whether $G \models \varphi$ in time $f(tw(G), |\varphi|)n$. Dynamic programming is only fast if the tables are small.

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

- Remove all colors except for the $\leq |\varphi|$ many that occur in $\varphi$.
- “Normalize” all formulas:
  $$\exists x \ x = x$$

Show the claim by induction.
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 $\xi$ 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 $\xi$ in $M_i(\xi)$.

- Remove all colors except for the $\leq |\varphi|$ many that occur in $\varphi$.
- “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.
Show the claim by induction. Base case $q = 0$: There are only $2^{2O(k^2 \cdot |\varphi|)}$ many quantifier-free formulas with $\leq k$ variables.
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
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 $1 \leq j \leq k$
Show the claim by induction. Base case $q = 0$: There are only $2^{O(k^2 \cdot |\varphi|)}$ 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 $1 \leq i, j \leq k$
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 $1 \leq i, j \leq k$
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 $2^{O(k^2 \cdot |\varphi|)}$ 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: $O(k^2 |\varphi|)$. 
Show the claim by induction. Base case $q = 0$: There are only $2^{O(k^2|\varphi|)}$ 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): $2^{O(k^2|\varphi|)}$.
Show the claim by induction. Base case $q = 0$: There are only $2^{O(k^2 \cdot |\varphi|)}$ 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): $2^{2^{O(k^2 |\varphi|)}}$
Next, we bound number of formulas with quantifier-rank \( \leq q \) and \( \leq k \) free variables.
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.
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

$$(\forall x \xi_1 \land \exists x \xi_4 \land \exists x \xi_8 \land \ldots) \lor$$
$$(\exists x \xi_3 \land \forall x \xi_2 \land \exists x \xi_9 \land \forall x \xi_1 \land \ldots) \lor$$
$$(\exists x \xi_5 \land \forall x \xi_8 \land \ldots) \lor \ldots$$

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

In total, the number of formulas is roughly $2 \cdot \cdots \cdot 2^{O(tw(G)^2 \cdot |\varphi|)}$.

This bound cannot be improved much.
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

$$\left( \forall x \xi_1 \land \exists x \xi_4 \land \exists x \xi_8 \land \ldots \right) \lor$$

$$\left( \exists x \xi_3 \land \forall x \xi_2 \land \exists x \xi_9 \land \forall x \xi_1 \land \ldots \right) \lor$$

$$\left( \exists x \xi_5 \land \forall x \xi_8 \land \ldots \right) \lor \ldots$$

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

$$(\forall x \xi_1 \land \exists x \xi_4 \land \exists x \xi_8 \land \ldots) \lor$$

$$(\exists x \xi_3 \land \forall x \xi_2 \land \exists x \xi_9 \land \forall x \xi_1 \land \ldots) \lor$$

$$(\exists x \xi_5 \land \forall x \xi_8 \land \ldots) \lor \ldots$$

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

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

$$(\forall x \xi_1 \land \exists x \xi_4 \land \exists x \xi_8 \land \ldots) \lor$$
$$(\exists x \xi_3 \land \forall x \xi_2 \land \exists x \xi_9 \land \forall x \xi_1 \land \ldots) \lor$$
$$(\exists x \xi_5 \land \forall x \xi_8 \land \ldots) \lor \ldots$$

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

$$2^{O(tw(G)^2 \cdot |\varphi|)}.$$ 

This bound cannot be improved much.
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.
$G[V_i]$ is the empty graph.
$G[V_i]$ is the empty graph.

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

$\{v_1, \ldots, v_k\}$

$\text{bag } i$

$\text{bag } j$

$\text{bag } l$
Join Nodes

$G[V_j] \cong F$

$M_j(\xi) \cong q\text{-type}(F; v_1, \ldots)$

$G[V_l] \cong H$

$M_l(\xi) \cong q\text{-type}(H; v_1, \ldots)$
Join Nodes

\[ G[V_i] \cong F \cup H \]

\[ M_i(\xi) \cong q\text{-type}(F \cup H; v_1, \ldots) \]

\[ G[V_j] \cong F \]

\[ M_j(\xi) \cong q\text{-type}(F; v_1, \ldots) \]

\[ G[V_l] \cong H \]

\[ M_l(\xi) \cong q\text{-type}(H; v_1, \ldots) \]
Introduce Nodes

Theorem (Fefermann–Vaught)

If we know \( \#q\)-type \((F; v_1, \ldots, v_k)\)
and \( \#q\)-type \((H; v_1, \ldots, v_k, v_{k+1})\),
then we can compute \( \#q\)-type \((F \cup H; v_1, \ldots, v_{k+1})\).
Introduce Nodes

\[ G[V_j] \cong F \]

\[ M_j(\xi) \cong q\text{-type}(F; v_1, \ldots, v_k) \]
Theorem (Fefermann–Vaught)

If we know
\[ \# q\text{-type}(F; v_1, \ldots, v_k) \]
then we can compute
\[ \# q\text{-type}(H; v_1, \ldots, v_{k+1}) \]

\[ G[V_j] \cong F \]
\[ M_j(\xi) \cong \]
\[ q\text{-type}(F; v_1, \ldots, v_k) \]
Introduce Nodes

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

\[ G[V_j] \cong F \]
\[ M_j(\xi) \cong q\text{-type}(F; v_1, \ldots, v_k) \]

\[ G[\{v_1, \ldots, v_{k+1}\}] \cong H \]

compute
\[ q\text{-type}(H; v_1, \ldots, v_{k+1}) \]
Theorem (Fefermann–Vaught)

If we know

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

then we can compute

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

\[ G[V_i] \models F \cup H \]
\[ M_i(\xi) \models q\text{-type}(F \cup H; v_1, \ldots, v_k+1) \]

\[ G[V_j] \models F \]
\[ M_j(\xi) \models q\text{-type}(F; v_1, \ldots, v_k) \]

\[ G[\{v_1, \ldots, v_k+1\}] \models H \]

compute

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

bag \(i\)

bag \(j\)
Introduce Nodes

\[ G[V_i] \cong F \cup H \]
\[ M_i(\xi) \cong \text{type}(F \cup H; v_1, \ldots, v_{k+1}) \]

**Theorem (Fefermann-Vaught)**

If we know

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

then we can compute

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

\[ G[\{v_1, \ldots, v_{k+1}\}] \cong H \]

compute

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

\[ G[V_j] \cong F \]
\[ M_j(\xi) \cong \text{q-type}(F; v_1, \ldots, v_k) \]
Forget Nodes

$G[V_i] = G[V_j]$

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

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

Remove all formulas that mention $x_k$. 

$\{v_1, \ldots, v_{k-1}\}$

$\{v_1, \ldots, v_k\}$

$\text{bag } i$

$\text{bag } j$
$G[V_i] = G[V_j]$
$G[V_i] = G[V_j]$  

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

$M_j(\xi)$
\[ G[V_i] = G[V_j] \]

\[ M_j(\xi) : \text{Formulas over } x_1, \ldots, x_k. \]

\[ M_i(\xi) : \text{Formulas over } x_1, \ldots, x_{k-1}. \]
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 \( x_k \).
For a $\text{MSO}_1$ formula $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time

$$\underbrace{2 \cdot \ldots \cdot 2}_{2^{O(tw(G)^2 |\varphi|)}}^{O(tw(G)^2 |\varphi|)} n.$$
For a MSO$_1$ formula $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time

$$2 \cdot 2^{O(tw(G)^2 |\varphi|)} 2^{n} \cdot 2|\varphi|$$

Can this be improved?
For a MSO$_1$ formula $\varphi$ and graph $G$ one can decide whether $G \models \varphi$ in time

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

Can this be improved?

No. It cannot be done without such a tower of powers (Frick, Grohe 2004).
Usually, Courcelle’s theorem is considered a theoretical classification tool.
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 …
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
-- 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))  
      )  
    )  
  )  
)
We saw the proof for the base variant $\text{MSO}_1$. 

By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant $\text{CMSO}$. 

Also we already know how to reduce $\text{MSO}_2$ to $\text{MSO}_1$. 

The proof for the optimization variant $\text{LinEMSOL}$ works similarly by also keeping track of the largest satisfying assignment.
Proof for Extensions

- We saw the proof for the base variant $\text{MSO}_1$.
- By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant CMSO.
We saw the proof for the base variant $\text{MSO}_1$.

By swapping in FV theorem for CMSO, we get the proof for the parity/modulo variant CMSO.

Also we already know how to reduce $\text{MSO}_2$ to $\text{MSO}_1$. 
Proof for Extensions

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

- On the other hand $\text{MSO}_2$ only holds for treewidth.
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 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

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

Courcelle’s Theorem
For a MSO$_1$ sentence $\varphi$ and graph G one can decide whether $G \models \varphi$ in time $f(cw(G), |\varphi|)n^3$ for some function $f$. 
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.
This approach can be extended to tree-like graphs (bounded treewidth).
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
- ...

[Diagram of tree-like graph]
How about planar graphs?
Independent Set on Planar Graphs

How about planar graphs?

INDEPENDENTSET is NP-complete on planar graphs.
One can decide whether a planar graph has an independent set of size $k$ in time $O(6^k n)$.

\begin{algorithm}
\textbf{IS}(G, k):
  
  if $G$ is empty return $k == 0$
  
  find vertex $v$ with degree $\leq 5$ in $G$
  
  for all $w \in N(v)$:
    
    if IS($G \setminus N(w)$, $k - 1$) return True
  
  return False
\end{algorithm}
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!
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.
A class $C$ has **bounded treewidth** if there exists a constant $c$ such that for all $G \in C$ 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 $G \in C$ holds $X \leq c$.

Assume we have a bounded treewidth class $C$. On this class, Coucelle’s theorem solves MSO$_1$ formulas in time $f(|\phi|, \text{tw}(G)) n \leq f(|\phi|, c)n = f'(|\phi|)n$. 

```bash
|graph_classes|
```

29
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!
A class $C$ has *bounded treewidth* if there exists a constant $c$ such that for all $G \in C$ 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 $G \in C$ 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 $\text{MSO}_1$ formulas in time $f(|\varphi|, \text{tw}(G))n \leq f(|\varphi|, c)n = f’(|\varphi|)n$. 
Courcelle’s Theorem

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

Let $\mathcal{C}$ be a graph class with bounded treewidth. There exists a function $f$ (depending on $\mathcal{C}$!) such that for every MSO$_1$ sentence $\varphi$ 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 $\varphi$, 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.
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.

Problems seem to be easier if the graphs are sparse!
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?
Every graph is “sparse” if you subdivide the edges.
Every graph is “sparse” if you subdivide the edges.

subdivision adds $1/2$ vertex per edge
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
Every graph is “sparse” if you subdivide the edges.

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

- Bounded expansion
- Excluding a topological minor
- Excluding a minor
- Bounded treewidth
- Bounded treedepth
- Star forests
- Planar
- Outerplanar
- Locally bounded treewidth
- Locally excluding a minor
- Bounded degree
- Linear forests
- Bounded genus
- Nowhere dense

Figure by Felix Reidl
For sparse graphs, $\text{MSO}_1$ 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.
For sparse graphs, $\text{MSO}_1$ is too powerful. For example Independent Set, Coloring, Dominating Set are NP-complete on planar graphs or bounded degree graph classes. However, \textit{first-order logic} fits just right.

Main Result (roughly)

Let $\mathcal{C}$ be a sparse graph class. For an FO formula $\varphi$ and graph $G \in \mathcal{C}$ one can decide whether $G \models \varphi$ in time $f(|\varphi|)n$ for some function $f$. 
For a given signature \( \tau \), first-order logic has ...

- element-variables \((x, y, z, \ldots)\)
- the equality relation \(=\) as well as the relations from \(\tau\).
- quantifiers \(\exists\) and \(\forall\), as well as operators \(\land\), \(\lor\) and \(\neg\).

We mostly work on colored undirected graphs with \(\tau = \{\sim, c_1, c_2, \ldots\}\). Here, we call the logic FO.
Can these properties be expressed in FO logic?
Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size $k$. 
  
  $\exists x_1 \ldots \exists x_k \neg x_i \sim x_j \land \neg x_i = x_j$
  
- There exists a dominating set of size $k$.
  
  $\exists x_1 \ldots \exists x_k \forall y \lor y \sim x_i \lor y = x_i$
  
- The number of vertices is even.
  
  No.
  
- The graph is connected.
  
  No.
Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.

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

- There exists a dominating set of size $k$.

- The number of vertices is even.

- No.

- The graph is connected.

- No.
Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.
  \[
  \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \land \neg x_i = x_j
  \]

- There exists a dominating set of size $k$. 

The number of vertices is even. No.

The graph is connected. No.
Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.

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

- There exists a dominating set of size $k$.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_i y \sim x_i \lor y = x_i$$
Can these properties be expressed in FO logic?

- There exists an independent set of size \( k \).
  \[
  \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \land \neg x_i = x_j
  \]

- There exists a dominating set of size \( k \).
  \[
  \exists x_1 \ldots \exists x_k \forall y \bigvee_i y \sim x_i \lor y = x_i
  \]

- The number of vertices is even.
Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.

\[
\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \land \neg x_i = x_j
\]

- There exists a dominating set of size $k$.

\[
\exists x_1 \ldots \exists x_k \forall y \bigvee_i y \sim x_i \lor y = x_i
\]

- The number of vertices is even. No.
Expressiveness

Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.

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

- There exists a dominating set of size $k$.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_i y \sim x_i \lor y = x_i$$

- The number of vertices is even. No.
- The graph is connected.
Can these properties be expressed in FO logic?

- There exists an independent set of size $k$.

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

- There exists a dominating set of size $k$.

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

- The number of vertices is even. No.
- The graph is connected. No.
Database languages such as SQL build upon first-order logic.
Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query ⇔
  Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$. 
Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query ⇔
  Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.
- Boolean query ⇔ Decide whether $G \models \varphi$. 
Database languages such as SQL build upon first-order logic.

- Enumerate all answers to a database query $\iff$ Enumerate all $v_1, \ldots, v_k$ with $G \models \varphi(v_1, \ldots, v_k)$.
- Boolean query $\iff$ Decide whether $G \models \varphi$.
- There exist extensions of first-order logic simulating SQL’s COUNT operator.
## First-Order Model-Checking (Query Evaluation)

**Input:** Graph $G$ and first-order sentence $\varphi$  
**Question:** $G \models \varphi$?
# Central Problems

## First-Order Model-Checking (Query Evaluation)

**Input:** Graph $G$ and first-order sentence $\varphi$

**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 $G \models \varphi(v_1, \ldots, v_k)$. 


### Central Problems

<table>
<thead>
<tr>
<th>First-Order Model-Checking (Query Evaluation)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph $G$ and first-order sentence $\varphi$</td>
</tr>
<tr>
<td><strong>Question:</strong> $G \models \varphi$?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First-Order Query Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph $G$ and first-order formula $\varphi(x_1, \ldots, x_k)$</td>
</tr>
</tbody>
</table>
| **Question:** Enumerate all $v_1, \ldots, v_k$ with  
$G \models \varphi(v_1, \ldots, v_k)$.

<table>
<thead>
<tr>
<th>First-Order Query Counting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph $G$ and first-order formula $\varphi(x_1, \ldots, x_k)$</td>
</tr>
</tbody>
</table>
| **Question:** Count number of tuples $v_1, \ldots, v_k$ with  
$G \models \varphi(v_1, \ldots, v_k)$.

40
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.

\[ \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \land \neg x_i = x_j \]
Theorem (Vardi 1982)

The model-checking problem is PSPACE-complete.

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

Proof: Reduction from Independent Set.

\[
\exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg x_i \sim x_j \land \neg x_i = x_j
\]

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

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

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 $k$.

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

- There exists a dominating set of size $k$.

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_i y \sim x_i \lor y = x_i$$

- The number of vertices is even. No.

- The graph is connected. No.
Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ such that $G \models \varphi$ iff $G$ is connected. How do we prove that?
Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ 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.
Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ 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 $\varphi$ 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 $G_q \equiv_q H_q$ using Ehrenfeucht–Fraïssé games.
Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ 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 $\varphi$ 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 $G_q \equiv_q H_q$. 

Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ 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 $\varphi$ 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.
Certain properties cannot be decided in first-order logic. For example, there is no first-order formula $\varphi$ 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 $\varphi$ 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*. 
The $q$-round Ehrenfeucht–Fraïssé game between the *Duplicator* and the *Spoiler* is played on two graphs $G$ and $H$.

Theorem

$G \equiv q H$ iff the Duplicator wins the $q$-round Ehrenfeucht–Fraïssé game.
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 $h_i \in H(G)$

---

$G$

$g_1$

$H$
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.
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).

**Theorem**

$G \equiv_H q$ if the Duplicator wins the $q$-round Ehrenfeucht–Fraïssé game.
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).

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

\[ G \]
\begin{center}
\begin{tikzpicture}

\node[vertex] (g1) at (0,0) {};
\node[vertex] (g2) at (1,0) {};
\node[vertex] (g3) at (2,0) {};
\node[vertex] (g4) at (3,0) {};
\node[vertex] (g5) at (4,0) {};
\node[vertex] (g6) at (5,0) {};
\node[vertex] (g7) at (6,0) {};
\node[vertex] (g8) at (7,0) {};
\node[vertex] (g9) at (8,0) {};
\node[vertex] (g10) at (9,0) {};

\draw (g1) -- (g2);
\draw (g2) -- (g3);
\draw (g3) -- (g4);
\draw (g4) -- (g5);
\draw (g5) -- (g6);
\draw (g6) -- (g7);
\draw (g7) -- (g8);
\draw (g8) -- (g9);
\draw (g9) -- (g10);
\end{tikzpicture}
\end{center}

\[ H \]
\begin{center}
\begin{tikzpicture}

\node[vertex] (h1) at (0,0) {};
\node[vertex] (h2) at (1,0) {};
\node[vertex] (h3) at (2,0) {};
\node[vertex] (h4) at (3,0) {};
\node[vertex] (h5) at (4,0) {};
\node[vertex] (h6) at (5,0) {};
\node[vertex] (h7) at (6,0) {};
\node[vertex] (h8) at (7,0) {};
\node[vertex] (h9) at (8,0) {};
\node[vertex] (h10) at (9,0) {};

\draw (h1) -- (h2);
\draw (h2) -- (h3);
\end{tikzpicture}
\end{center}
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.
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 $i, j$. 

\[ G \]
\[ H \]
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.
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
\[
\forall i, j \quad g_i \sim g_j \iff h_i \sim h_j
\]

Theorem \( G \equiv_q H \) iff the Duplicator wins the \( q \)-round Ehrenfeucht–Fraïssé game.
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 $i, j$. 

![Diagram of graphs G and H](image)
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.
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.
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.
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 $G \equiv_H q \iff$ the Duplicator wins the $q$-round Ehrenfeucht–Fraïssé game.
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.
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.
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.
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.