Introduction to logic in graphs and algorithmic meta-theorems

Ignasi Sau

LIRMM, Université de Montpellier, CNRS, France

20th JCALM

LIRMM, Montpellier, December 13-14, 2023

20èmes JCALM

JCALM: Journées de Combinatoire et d'Algorithmes du Littoral Méditerranéen.

Take place once or twice a year since 2006.

Involved research teams:

- DALGO and ACRO teams at LIS in Marseille.
- AIGCo team (among others) at LIRMM in Montpellier.
- COATI common project at I3S and INRIA Sophia-Antipolis.
- GAPCOMB team at UPC in Barcelona.

Current JCALM

Topic: Logic and graph algorithms

Programme

Wednesday December 13th

- 09:30–10:20: Café+croissants et accueil des participant-e-s (cafétéria du LIRMM, bâtiment 4)
- 10:20–11:20: **Ignasi Sau**, Introduction to logic in graphs and algorithmic meta-theorems
- 11:30–12:30: **Hugo Jacob**, First-order model-checking on sparse graph classes
- 12:30–14:00: Repas (cafétéria du LIRMM, bâtiment 4)
- 14:00–15:00: **Eunjung Kim**, First-order model-checking on graphs of bounded twin-width
- 15:10–16:10: **Dimitrios Thilikos**, Logic and algorithms for graph minors
- 16:10–16:40: Café+croissants
- 16:40–17:30: Open problems
- Soirée: dîner en centre ville de Montpellier

Thursday December 14th

- 09:30–10:20: Café+croissants (cafétéria du LIRMM, bâtiment 4)
- 10:20–11:20: **Matthieu Rosenfeld**, Monadic second-order logic and treewidth
- 11:30–12:30: **Giannos Stamoulis**, Elementary first-order model-checking
- 12:00–14:00: Repas (cafétéria du LIRMM, bâtiment 4)

Algorithmic meta-theorems

Typical statement of an algorithmic meta-theorem (AMT):

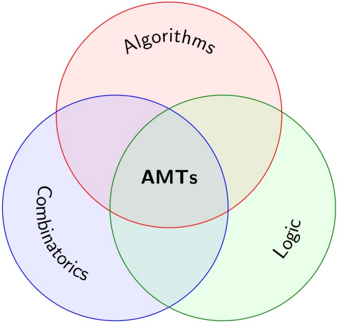
> Every computational problem that can be formalized in a given logic $\mathcal{L}$ can be solved efficiently on every class $\mathcal{C}$ of structures (typically, graphs) satisfying certain (typically, combinatorial) conditions.

Algorithmic meta-theorems

Typical statement of an algorithmic meta-theorem (AMT):

> Every computational problem that can be formalized in a given logic $\mathcal{L}$ can be solved efficiently on every class $\mathcal{C}$ of structures (typically, graphs) satisfying certain (typically, combinatorial) conditions.

- Logical component: given by a logic $\mathcal{L}$ (such as first-order or second-order logic).
- Structural (combinatorial) component: given by a class $\mathcal{C}$ (such as planar graphs, or graphs of bounded degree).



Outline of this introductory talk

★ Strongly inspired from the survey of Stephan Kreutzer (2011):

"Algorithmic Meta-Theorems"

1. Introduction to logic (in graphs)
2. AMTs for monadic second-order logic
3. AMTs for first-order logic

Next section is...

1. Introduction to logic (in graphs)
2. AMTs for monadic second-order logic
3. AMTs for first-order logic

Basics on logic

Signature $\sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_q\}$: finite set of relation symbols R_i and constant symbols c_i.

Example of a relation symbol: "edge" relation in graphs, with arity 2.

Basics on logic

Signature $\sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_q\}$: finite set of relation symbols R_i and constant symbols c_i.

Example of a relation symbol: "edge" relation in graphs, with arity 2.

Sometimes a signature is also called a vocabulary.

Basics on logic

Signature $\sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_q\}$: finite set of relation symbols R_i
and constant symbols c_i.

Example of a relation symbol: "edge" relation in graphs, with arity 2.

Sometimes a signature is also called a vocabulary.

σ-structure $A = (V(A), R_1(A), \ldots, R_k(A), c_1(A), \ldots, c_q(A))$, such that:

- $V(A)$ is the universe (in graphs: vertex set).

Basics on logic

Signature $\sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_q\}$: finite set of relation symbols R_i and constant symbols c_i.

Example of a relation symbol: "edge" relation in graphs, with arity 2.

Sometimes a signature is also called a vocabulary.

σ-structure $A = (V(A), R_1(A), \ldots, R_k(A), c_1(A), \ldots, c_q(A))$, such that:

- $V(A)$ is the universe (in graphs: vertex set).
- For each $R_i \in \sigma$ of arity r, we have $R_i(A) \subseteq V(A)^r$ (example: edges).

Basics on logic

Signature $\sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_q\}$: finite set of relation symbols R_i and constant symbols c_i.

Example of a relation symbol: "edge" relation in graphs, with arity 2.

Sometimes a signature is also called a vocabulary.

σ-structure $A = (V(A), R_1(A), \ldots, R_k(A), c_1(A), \ldots, c_q(A))$, such that:

- $V(A)$ is the universe (in graphs: vertex set).
- For each $R_i \in \sigma$ of arity r, we have $R_i(A) \subseteq V(A)^r$ (example: edges).
- For each $c_i \in \sigma$, we have a constant $c_i(A) \in V(A)$ (i.e., a vertex)

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ FO[σ]: class of formulas of first-order logic over σ:

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ FO[σ]: class of formulas of first-order logic over σ:

- If $R \in \sigma$ and $\overline{x}$ is a tuple of σ-terms of length ar(R), then $R\overline{x} \in$ FO[σ] (in graphs, for the edge relation: adjacency).

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ $\mathrm{FO}[\sigma]$: class of formulas of first-order logic over σ:

- If $R \in \sigma$ and $\overline{x}$ is a tuple of σ-terms of length $\mathrm{ar}(R)$, then $R\overline{x} \in \mathrm{FO}[\sigma]$ (in graphs, for the edge relation: adjacency).
- If t and s are σ-terms then $t = s \in \mathrm{FO}[\sigma]$.

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ FO[σ]: class of formulas of first-order logic over σ:

- If $R \in \sigma$ and $\overline{x}$ is a tuple of σ-terms of length $\mathrm{ar}(R)$, then $R\overline{x} \in \mathrm{FO}[\sigma]$ (in graphs, for the edge relation: adjacency).
- If t and s are σ-terms then $t = s \in \mathrm{FO}[\sigma]$.
- If $\varphi, \psi \in \mathrm{FO}[\sigma]$, then so are $\varphi \vee \psi$, $\varphi \wedge \psi$, $\neg\varphi$.

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ FO[σ]: class of formulas of first-order logic over σ:

- If $R \in \sigma$ and $\overline{x}$ is a tuple of σ-terms of length $\mathrm{ar}(R)$, then $R\overline{x} \in \mathrm{FO}[\sigma]$ (in graphs, for the edge relation: adjacency).
- If t and s are σ-terms then $t = s \in \mathrm{FO}[\sigma]$.
- If $\varphi, \psi \in \mathrm{FO}[\sigma]$, then so are $\varphi \vee \psi$, $\varphi \wedge \psi$, $\neg\varphi$.
- If $\varphi \in \mathrm{FO}[\sigma]$ and x is a first-order variable, then $\exists x \varphi \in \mathrm{FO}[\sigma]$ and $\forall x \varphi \in \mathrm{FO}[\sigma]$.

First-order and monadic second-order logic

▷ Let σ be a signature (example: edges, for graphs).

▷ We assume a countably infinite set of first-order variables $x, y, \ldots$ (elements) and second-order variables $X, Y, \ldots$ (sets of elements).

▷ σ-term: first-order variable or constant symbol $c \in \sigma$.

▷ $\mathrm{FO}[\sigma]$: class of formulas of first-order logic over σ:

- If $R \in \sigma$ and $\overline{x}$ is a tuple of σ-terms of length $\mathrm{ar}(R)$, then $R\overline{x} \in \mathrm{FO}[\sigma]$ (in graphs, for the edge relation: adjacency).
- If t and s are σ-terms then $t = s \in \mathrm{FO}[\sigma]$.
- If $\varphi, \psi \in \mathrm{FO}[\sigma]$, then so are $\varphi \vee \psi$, $\varphi \wedge \psi$, $\neg\varphi$.
- If $\varphi \in \mathrm{FO}[\sigma]$ and x is a first-order variable, then $\exists x \varphi \in \mathrm{FO}[\sigma]$ and $\forall x \varphi \in \mathrm{FO}[\sigma]$.

▷ $\mathrm{MSO}[\sigma]$: class of formulas of monadic second-order logic over σ:

- Additional rule: if X is a second-order variable and $\varphi \in \mathrm{MSO}[\sigma]$, then $\exists X \varphi \in \mathrm{MSO}[\sigma]$ and $\forall X \varphi \in \mathrm{MSO}[\sigma]$.

Some more notation

▷ We define:

- $\text{FO} = \bigcup_\sigma \text{FO}[\sigma]$
- $\text{MSO} = \bigcup_\sigma \text{MSO}[\sigma]$

Some more notation

▷ We define:

- FO $= \bigcup_\sigma \mathsf{FO}[\sigma]$
- MSO $= \bigcup_\sigma \mathsf{MSO}[\sigma]$

▷ Usual notation:

- Connectors: $=$ (equality), $\vee$ (conjunction), $\wedge$ (disjunction), $\neg$ (negation).
- If $\overline{x}$ is a tuple and R a relation, $R\overline{x}$ denotes containment $(\in, \subseteq)$ in R.
- Quantifiers: $\exists$ (existential) and $\forall$ (universal).

Some more notation

▷ We define:

- FO $= \bigcup_\sigma \mathrm{FO}[\sigma]$
- MSO $= \bigcup_\sigma \mathrm{MSO}[\sigma]$

▷ Usual notation:

- Connectors: $=$ (equality), $\vee$ (conjunction), $\wedge$ (disjunction), $\neg$ (negation).
- If $\overline{x}$ is a tuple and R a relation, $R\overline{x}$ denotes containment $(\in, \subseteq)$ in R.
- Quantifiers: $\exists$ (existential) and $\forall$ (universal).

▷ Abbreviations:

- $x \neq y$, instead of $\neg x = y$.
- $\varphi \rightarrow \psi$ instead of $(\neg\varphi \vee \psi)$.

Free variables and models

▷ Free variables of a formula: those that are not involved in any quantifier. Denoted $\varphi(\overline{x})$.

Free variables and models

▷ Free variables of a formula: those that are not involved in any quantifier. Denoted $\varphi(\overline{x})$.

▷ Sentence: formula with no free variables.

Free variables and models

- Free variables of a formula: those that are not involved in any quantifier. Denoted $\varphi(\overline{x})$.

- Sentence: formula with no free variables.

- Notation $A \models \varphi$: "A satisfies φ" or "A is a model of φ".

Free variables and models

▷ Free variables of a formula: those that are not involved in any quantifier. Denoted $\varphi(\overline{x})$.

▷ Sentence: formula with no free variables.

▷ Notation $A \models \varphi$: "A satisfies φ" or "A is a model of φ".

▷ If $\varphi(\overline{x})$ has free variables $\overline{x}$, and $\overline{a}$ is a tuple of the same length as $\overline{x}$, we write $A \models \varphi(\overline{a})$ or $(A, \overline{a}) \models \varphi$ if φ is true when $\overline{x}$ is interpreted as $\overline{a}$.

Free variables and models

- Free variables of a formula: those that are not involved in any quantifier. Denoted $\varphi(\overline{x})$.
- Sentence: formula with no free variables.
- Notation $A \models \varphi$: "A satisfies φ" or "A is a model of φ".
- If $\varphi(\overline{x})$ has free variables $\overline{x}$, and $\overline{a}$ is a tuple of the same length as $\overline{x}$, we write $A \models \varphi(\overline{a})$ or $(A, \overline{a}) \models \varphi$ if φ is true when $\overline{x}$ is interpreted as $\overline{a}$.
- If we deal with (non-annotated) graphs: $\sigma = E$ (i.e., the edge relation).

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \ldots \exists x_k \bigwedge_{1 \le i < j \le k} \left(x_i \neq x_j \wedge \neg E x_i x_j \right)$$

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \dots \exists x_k \bigwedge_{1 \le i < j \le k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

A graph $G \models \varphi_k$ if and only if G contains an independent set (set of pairwise non-adjacent vertices) of size k.

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \land \neg E x_i x_j)$$

A graph $G \models \varphi_k$ if and only if G contains an independent set (set of pairwise non-adjacent vertices) of size k.

$$\varphi(X) := \forall x (Xx \lor \exists z (Exz \land Xz))$$

In this formula, X is a free variable.

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \dots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \land \neg E x_i x_j)$$

A graph $G \models \varphi_k$ if and only if G contains an independent set (set of pairwise non-adjacent vertices) of size k.

$$\varphi(X) := \forall x (Xx \lor \exists z (Exz \land Xz))$$

In this formula, X is a free variable.
A pair (G, U), where $U \subseteq V(G)$, satisfies $(G, U) \models \varphi$ if and only if U is a dominating set in G (every vertex not in U has a neighbor in U).

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \dots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

A graph $G \models \varphi_k$ if and only if G contains an independent set (set of pairwise non-adjacent vertices) of size k.

$$\varphi(X) := \forall x (Xx \vee \exists z (Exz \wedge Xz))$$

In this formula, X is a free variable.
A pair (G, U), where $U \subseteq V(G)$, satisfies $(G, U) \models \varphi$ if and only if U is a dominating set in G (every vertex not in U has a neighbor in U).

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

Examples of FO formulas in graphs

$$\varphi_k := \exists x_1 \dots \exists x_k \bigwedge_{1 \le i < j \le k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

A graph $G \models \varphi_k$ if and only if G contains an independent set (set of pairwise non-adjacent vertices) of size k.

$$\varphi(X) := \forall x (Xx \vee \exists z (Exz \wedge Xz))$$

In this formula, X is a free variable.
A pair (G, U), where $U \subseteq V(G)$, satisfies $(G, U) \models \varphi$ if and only if U is a dominating set in G (every vertex not in U has a neighbor in U).

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

Expresses that a graph contains a dominating set of size k.

Independent set versus dominating set

- Independent set of size k:

$$\exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

- Dominating set of size k:

$$\exists x_1 \ldots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

Independent set versus dominating set

- Independent set of size k:

$$\exists x_1 \dots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

- Dominating set of size k:

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

Independent set versus dominating set

- Independent set of size k:

$$\exists x_1 \dots \exists x_k \bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

- Dominating set of size k:

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

The second formula has an alternation of quantifiers.

Independent set versus dominating set

- Independent set of size k:

$$\exists x_1 \dots \exists x_k \bigwedge_{1 \le i < j \le k} (x_i \neq x_j \wedge \neg E x_i x_j)$$

- Dominating set of size k:

$$\exists x_1 \dots \exists x_k \forall y \bigvee_{i=1}^{k} (y = x_i \vee E x_i y)$$

The second formula has an alternation of quantifiers.

This suggests that the DOMINATING SET problem might be harder than the INDEPENDENT SET problem, as we shall see later...

Examples of MSO formulas in graphs

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x (X_1 x \vee X_2 x) \wedge (X_1 x \to \neg X_2 x) \wedge (X_2 x \to \neg X_1 x)$$

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x (X_1 x \vee X_2 x) \wedge (X_1 x \rightarrow \neg X_2 x) \wedge (X_2 x \rightarrow \neg X_1 x)$$

$(G, U_1, U_2) \models \varphi$ if and only if (U_1, U_2) is a bipartition of $V(G)$.

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x(X_1x \vee X_2x) \wedge (X_1x \to \neg X_2x) \wedge (X_2x \to \neg X_1x)$$

$(G, U_1, U_2) \models \varphi$ if and only if (U_1, U_2) is a bipartition of $V(G)$.

$$\forall X_1 \forall X_2((\varphi(X_1, X_2)) \wedge (\exists y_1 \exists y_2(X_1y_1 \wedge X_2y_2)) \to (\exists x_1 \exists x_2(X_1x_1 \wedge X_2x_2 \wedge Ex_1x_2)))$$

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x (X_1 x \vee X_2 x) \wedge (X_1 x \rightarrow \neg X_2 x) \wedge (X_2 x \rightarrow \neg X_1 x)$$

$(G, U_1, U_2) \models \varphi$ if and only if (U_1, U_2) is a bipartition of $V(G)$.

$$\forall X_1 \forall X_2 ((\varphi(X_1, X_2)) \wedge (\exists y_1 \exists y_2 (X_1 y_1 \wedge X_2 y_2)) \rightarrow (\exists x_1 \exists x_2 (X_1 x_1 \wedge X_2 x_2 \wedge E x_1 x_2)))$$

If ψ is this formula, a graph $G \models \psi$ if and only if G is connected.

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x (X_1 x \vee X_2 x) \wedge (X_1 x \to \neg X_2 x) \wedge (X_2 x \to \neg X_1 x)$$

$(G, U_1, U_2) \models \varphi$ if and only if (U_1, U_2) is a bipartition of $V(G)$.

$$\forall X_1 \forall X_2 ((\varphi(X_1, X_2)) \wedge (\exists y_1 \exists y_2 (X_1 y_1 \wedge X_2 y_2)) \to (\exists x_1 \exists x_2 (X_1 x_1 \wedge X_2 x_2 \wedge E x_1 x_2)))$$

If ψ is this formula, a graph $G \models \psi$ if and only if G is connected.

$$\tau := \exists C_1 \exists C_2 \exists C_3 (\forall x \bigvee_{i=1}^{3} C_i x) \wedge \forall x \forall y (Exy \to \bigwedge_{i=1}^{3} \neg (C_i x \vee C_i y))$$

Examples of MSO formulas in graphs

$$\varphi(X_1, X_2) := \forall x(X_1 x \vee X_2 x) \wedge (X_1 x \rightarrow \neg X_2 x) \wedge (X_2 x \rightarrow \neg X_1 x)$$

$(G, U_1, U_2) \models \varphi$ if and only if (U_1, U_2) is a bipartition of $V(G)$.

$$\forall X_1 \forall X_2((\varphi(X_1, X_2)) \wedge (\exists y_1 \exists y_2(X_1 y_1 \wedge X_2 y_2)) \rightarrow (\exists x_1 \exists x_2(X_1 x_1 \wedge X_2 x_2 \wedge E x_1 x_2)))$$

If ψ is this formula, a graph $G \models \psi$ if and only if G is connected.

$$\tau := \exists C_1 \exists C_2 \exists C_3(\forall x \bigvee_{i=1}^{3} C_i x) \wedge \forall x \forall y(Exy \rightarrow \bigwedge_{i=1}^{3} \neg(C_i x \vee C_i y))$$

A graph $G \models \tau$ if and only if G is 3-colorable.

The model-checking problem

Let $\mathcal{L}$ be a fixed logic. We define the Model-Checking problem of $\mathcal{L}$:

$\mathrm{MC}(\mathcal{L})$
Input: A structure A and a sentence $\varphi \in \mathcal{L}$.
Question: $A \models \varphi$?

The model-checking problem

Let $\mathcal{L}$ be a fixed logic. We define the MODEL-CHECKING problem of $\mathcal{L}$:

MC($\mathcal{L}$)
Input: A structure A and a sentence $\varphi \in \mathcal{L}$.
Question: $A \models \varphi$?

Related problem:

SATISFIABILITY($\mathcal{L}$)
Input: A sentence $\varphi \in \mathcal{L}$.
Question: Does there exist a structure A such that $A \models \varphi$?

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* Quantified Boolean Satisfiability, *using two different elements of the structure to simulate the assignments "true" and "false".*

Particular case: FO model-checking

MC(FO)
Input: A structure A and a sentence $\varphi \in$ FO.
Question: $A \models \varphi$?

- Bad news: MC(FO) is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* Quantified Boolean Satisfiability, *using two different elements of the structure to simulate the assignments "true" and "false".*

- Good news: Solvable in polynomial time for every fixed formula.

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$ **Input:** A structure A and a sentence $\varphi \in \mathsf{FO}$. **Question:** $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* QUANTIFIED BOOLEAN SATISFIABILITY*, using two different elements of the structure to simulate the assignments "true" and "false".*

- Good news: Solvable in polynomial time for every fixed formula.

  Idea: *try all possible choices for the first-order variables appearing in the formula, and check whether it is satisfied. Running time:* $|A|^{f(\varphi)}$.

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* QUANTIFIED BOOLEAN SATISFIABILITY, *using two different elements of the structure to simulate the assignments* "true" *and* "false".

- Good news: Solvable in polynomial time for every fixed formula.

  Idea: *try all possible choices for the first-order variables appearing in the formula, and check whether it is satisfied. Running time:* $|A|^{f(\varphi)}$.

The hardness depends on whether the formula is part of the input or not.

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* QUANTIFIED BOOLEAN SATISFIABILITY, *using two different elements of the structure to simulate the assignments "true" and "false".*

- Good news: Solvable in polynomial time for every fixed formula.

  Idea: *try all possible choices for the first-order variables appearing in the formula, and check whether it is satisfied. Running time:* $|A|^{f(\varphi)}$.

The hardness depends on whether the formula is part of the input or not.

Is it the end of the story?

Particular case: FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A structure A and a sentence $\varphi \in \mathsf{FO}$.
Question: $A \models \varphi$?

- Bad news: $\mathrm{MC}(\mathsf{FO})$ is PSPACE-complete, even restricted to structures with only 2 elements. [Vardi. 1982]

  Idea: *reduction from* QUANTIFIED BOOLEAN SATISFIABILITY, *using two different elements of the structure to simulate the assignments "true" and "false".*

- Good news: Solvable in polynomial time for every fixed formula.

  Idea: *try all possible choices for the first-order variables appearing in the formula, and check whether it is satisfied. Running time:* $|A|^{f(\varphi)}$.

The hardness depends on whether the formula is part of the input or not.

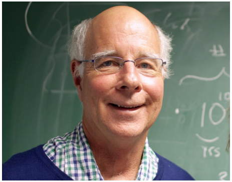
Is it the end of the story?

We need to parameterize the problem!

The area of parameterized complexity

Idea Measure the complexity of an algorithm in terms of the input size and an additional integer parameter.

This theory started in the late 80's, by Downey and Fellows:



Today, it is a well-established area with hundreds of articles published every year in the most prestigious TCS journals and conferences.

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$,
where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

- k-VERTEX COVER: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, containing at least an endpoint of every edge?
- INDEPENDENT SET: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \geq k$, of pairwise non-adjacent vertices?
- VERTEX k-COLORING: Can the vertices of a graph be colored with $\leq k$ colors, so that any two adjacent vertices get different colors?

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

- k-Vertex Cover: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, containing at least an endpoint of every edge?
- Independent Set: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \geq k$, of pairwise non-adjacent vertices?
- Vertex k-Coloring: Can the vertices of a graph be colored with $\leq k$ colors, so that any two adjacent vertices get different colors?

These three problems are NP-hard, but are they equally hard?

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n))$
- Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k)$
- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

- Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- *k*-VERTEX COVER: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- INDEPENDENT SET: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

  The problem is XP (slice-wise polynomial)

- VERTEX *k*-COLORING: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

  The problem is XP (slice-wise polynomial)

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

  The problem is para-NP-hard

Why k-Independent Set may not be FPT?

k-Independent Set: Solvable in time $k^2 \cdot n^k \;=\; \boxed{f(k) \cdot n^{g(k)}}$

Why k-Independent Set may not be FPT?

k-Independent Set: Solvable in time $k^2 \cdot n^k = \boxed{f(k) \cdot n^{g(k)}}$

Why k-Independent Set may not be FPT?

Why k-Independent Set may not be FPT?

k-Independent Set: Solvable in time $k^2 \cdot n^k \;=\; \boxed{f(k) \cdot n^{g(k)}}$

Why k-Independent Set may not be FPT?

So far, nobody has managed to find an FPT algorithm for k-Indep. Set.

(also, nobody has found a poly-time algorithm for SAT)

Why k-Independent Set may not be FPT?

k-Independent Set: Solvable in time $k^2 \cdot n^k \;=\; \boxed{f(k) \cdot n^{g(k)}}$

Why k-Independent Set may not be FPT?

So far, nobody has managed to find an FPT algorithm for k-Indep. Set.

(also, nobody has found a poly-time algorithm for SAT)

Working hypothesis of paramet. complexity: **k-Indep. Set is not FPT**

(in classical complexity: SAT cannot be solved in poly-time)

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

A parameterized reduction from A to B is an algorithm such that:

Instance (x, k) of A → time $f(k) \cdot |x|^{\mathcal{O}(1)}$ → Instance (x', k') of B

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

A parameterized reduction from A to B is an algorithm such that:

Instance (x, k) of A → time $f(k) \cdot |x|^{\mathcal{O}(1)}$ → Instance (x', k') of B

1. (x, k) is a YES-instance of $A \Leftrightarrow (x', k')$ is a YES-instance of B.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

A parameterized reduction from A to B is an algorithm such that:

Instance (x, k) of A → time $f(k) \cdot |x|^{\mathcal{O}(1)}$ → Instance (x', k') of B

1. (x, k) is a YES-instance of $A \Leftrightarrow (x', k')$ is a YES-instance of B.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

W[1]-hard problem: $\exists$ parameterized reduction from k-INDEP. SET to it.

W[2]-hard problem: $\exists$ param. reduction from k-DOMINATING SET to it.

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

A parameterized reduction from A to B is an algorithm such that:

Instance (x, k) of A → time $f(k) \cdot |x|^{\mathcal{O}(1)}$ → Instance (x', k') of B

1. (x, k) is a YES-instance of $A \Leftrightarrow (x', k')$ is a YES-instance of B.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

W[1]-hard problem: $\exists$ parameterized reduction from k-INDEP. SET to it.

W[2]-hard problem: $\exists$ param. reduction from k-DOMINATING SET to it.

W[i]-hard: strong evidence of not being FPT.

How to transfer hardness among parameterized problems?

Let A, B be two parameterized problems.

A parameterized reduction from A to B is an algorithm such that:

Instance (x, k) of A → time $f(k) \cdot |x|^{\mathcal{O}(1)}$ → Instance (x', k') of B

1. (x, k) is a YES-instance of $A \Leftrightarrow (x', k')$ is a YES-instance of B.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

W[1]-hard problem: $\exists$ parameterized reduction from k-INDEP. SET to it.

W[2]-hard problem: $\exists$ param. reduction from k-DOMINATING SET to it.

W[i]-hard: strong evidence of not being FPT. Hypothesis: FPT $\neq$ W[1]

Parameterized model-checking

Parameterized model-checking

The "right" problem to consider is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A structure A and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi|$.
Question: $A \models \varphi$?

Parameterized model-checking

The “right” problem to consider in graphs is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

Parameterized model-checking

The "right" problem to consider in graphs is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

k-Indep. Set is W[1]-hard

Parameterized model-checking

The "right" problem to consider in graphs is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $\lvert\varphi\rvert$.
Question: $G \models \varphi$?

k-Indep. Set is W[1]-hard $\Rightarrow$ $\mathrm{MC}(\mathrm{FO})$ is W[1]-hard (and XP).

Parameterized model-checking

The “right” problem to consider in graphs is the following :

MC($\mathcal{L}$)
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

We restrict the input graph G to belong to some particular graph class $\mathcal{C}$.

Parameterized model-checking

The "right" problem to consider in graphs is the following :

> $\mathrm{MC}(\mathcal{L})$
> **Input:** A graph G and a sentence $\varphi \in \mathcal{L}$.
> **Parameter:** $|\varphi|$.
> **Question:** $G \models \varphi$?

We restrict the input graph G to belong to some particular graph class $\mathcal{C}$.

> $\mathrm{MC}(\mathcal{L}, \mathcal{C})$
> **Input:** A graph $G \in \mathcal{C}$ and a sentence $\varphi \in \mathcal{L}$.
> **Parameter:** $|\varphi|$.
> **Question:** $G \models \varphi$?

Parameterized model-checking

The "right" problem to consider in graphs is the following :

> $\mathrm{MC}(\mathcal{L})$
> **Input:** A graph G and a sentence $\varphi \in \mathcal{L}$.
> **Parameter:** $|\varphi|$.
> **Question:** $G \models \varphi$?

We restrict the input graph G to belong to some particular graph class $\mathcal{C}$.

> $\mathrm{MC}(\mathcal{L}, \mathcal{C})$
> **Input:** A graph $G \in \mathcal{C}$ and a sentence $\varphi \in \mathcal{L}$.
> **Parameter:** $|\varphi|$.
> **Question:** $G \models \varphi$?

Holy grail: for which $\mathcal{L}$ and $\mathcal{C}$ is $\mathrm{MC}(\mathcal{L}, \mathcal{C})$ FPT?

Parameterized model-checking

The "right" problem to consider in graphs is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $\lvert\varphi\rvert$.
Question: $G \models \varphi$?

We restrict the input graph G to belong to some particular graph class $\mathcal{C}$.

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$
Input: A graph $G \in \mathcal{C}$ and a sentence $\varphi \in \mathcal{L}$.
Parameter: $\lvert\varphi\rvert$.
Question: $G \models \varphi$?

Holy grail: for which $\mathcal{L}$ and $\mathcal{C}$ is $\mathrm{MC}(\mathcal{L}, \mathcal{C})$ FPT? $f(\lvert\varphi\rvert) \cdot \lvert G\rvert^{\mathcal{O}(1)}$

Parameterized model-checking

The "right" problem to consider in graphs is the following :

$\mathrm{MC}(\mathcal{L})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

We restrict the input graph G to belong to some particular graph class $\mathcal{C}$.

$\mathrm{MC}(\mathcal{L}, \mathbf{p})$
Input: A graph G and a sentence $\varphi \in \mathcal{L}$.
Parameter: $|\varphi| + \mathbf{p}(G)$.
Question: $G \models \varphi$?

Holy grail: for which $\mathcal{L}$ and $\mathcal{C}$ is $\mathrm{MC}(\mathcal{L}, \mathcal{C})$ FPT?

$f(|\varphi|, \mathbf{p}(G)) \cdot |G|^{\mathcal{O}(1)}$

Next section is...

1. Introduction to logic (in graphs)
2. AMTs for monadic second-order logic
3. AMTs for first-order logic

MSO model-checking

$\mathrm{MC}(\mathsf{MSO})$
Input: A graph G and a sentence $\varphi \in \mathsf{MSO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

MSO model-checking

MC(MSO)
Input: A graph G and a sentence $\varphi \in$ MSO.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

3-Colorability is MSO-definable and NP-complete.

MSO model-checking

MC(MSO) **Input:** A graph G and a sentence $\varphi \in$ MSO. **Parameter:** $\lvert\varphi\rvert$. **Question:** $G \models \varphi$?

3-Colorability is MSO-definable and NP-complete.

MC(MSO) is para-NP-hard.

MSO model-checking

$\mathrm{MC}(\mathsf{MSO})$
Input: A graph G and a sentence $\varphi \in \mathsf{MSO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

3-Colorability is MSO-definable and NP-complete.

$\mathrm{MC}(\mathsf{MSO})$ is para-NP-hard.

$\mathrm{MC}(\mathsf{MSO}, \mathbf{p})$
Input: A graph G and a sentence $\varphi \in \mathsf{MSO}$.
Parameter: $|\varphi| + \mathbf{p}(G)$.
Question: $G \models \varphi$?

MSO model-checking

$\mathrm{MC}(\mathsf{MSO})$
Input: A graph G and a sentence $\varphi \in \mathsf{MSO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

3-Colorability is MSO-definable and NP-complete.

$\mathrm{MC}(\mathsf{MSO})$ is para-NP-hard.

$\mathrm{MC}(\mathsf{MSO}, \mathbf{p})$
Input: A graph G and a sentence $\varphi \in \mathsf{MSO}$.
Parameter: $|\varphi| + \mathbf{p}(G)$.
Question: $G \models \varphi$?

As some of you may imagine: we take $\mathbf{p} =$ treewidth.

Crucial notion: treewidth

- 1972: Bertelè and Brioschi (dimension).
- 1976: Halin (S-functions of graphs).
- 1984: Arnborg and Proskurowski (partial k-trees).
- 1984: Robertson and Seymour (treewidth).

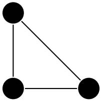
Crucial notion: treewidth

- 1972: Bertelè and Brioschi (dimension).
- 1976: Halin (*S*-functions of graphs).
- 1984: Arnborg and Proskurowski (partial *k*-trees).
- 1984: Robertson and Seymour (treewidth).

Treewidth measures the (topological) similarity of a graph with a forest.

Treewidth via k-trees

Example of a 2-tree:



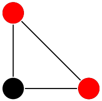
[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

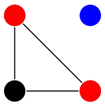
For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.



[Figure by Julien Baste]

Treewidth via k-trees

Example of a 2-tree:

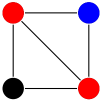


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

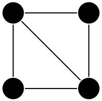


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

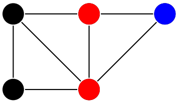


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

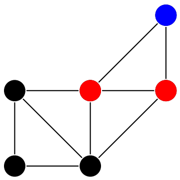


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

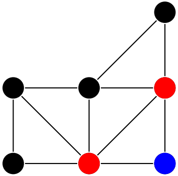


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

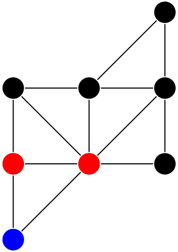


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

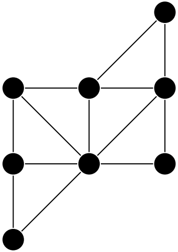


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

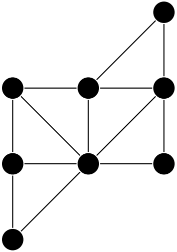


[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:



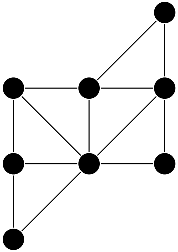
[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

A partial k-tree is a subgraph of a k-tree.

Treewidth via k-trees

Example of a 2-tree:



[Figure by Julien Baste]

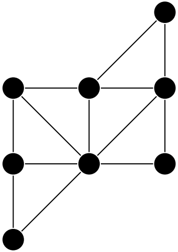
For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

A partial k-tree is a subgraph of a k-tree.

Treewidth of a graph G, denoted $\text{tw}(G)$: smallest integer k such that G is a partial k-tree.

Treewidth via k-trees

Example of a 2-tree:



[Figure by Julien Baste]

For $k \geq 1$, a k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

A partial k-tree is a subgraph of a k-tree.

Treewidth of a graph G, denoted $\text{tw}(G)$: smallest integer k such that G is a partial k-tree.

Construction suggests the notion of tree decomposition: small separators.

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G)\ \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:

  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.

An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

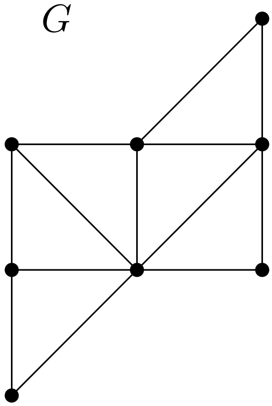
  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:

  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \ \ \forall t \in V(T)$ (bags),

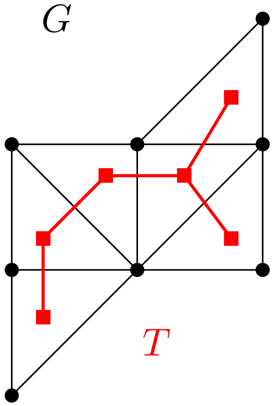
  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:

  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \ \ \forall t \in V(T)$ (bags),

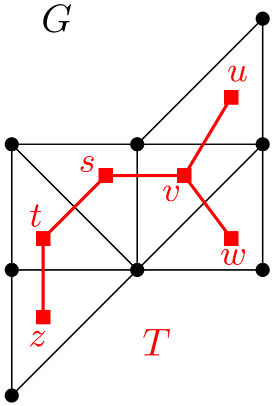
  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:

  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

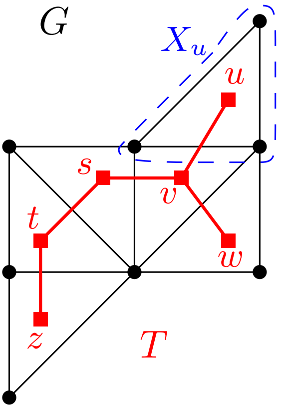
- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \ \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.
- Treewidth of a graph G, $\text{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

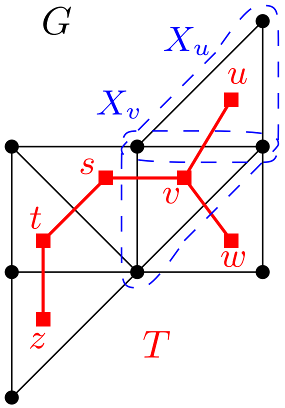
  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \ \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

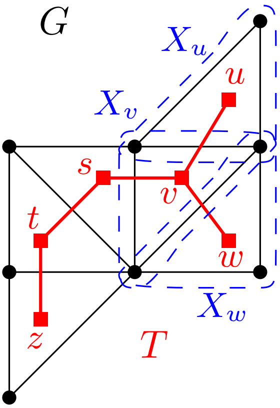
- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.
- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

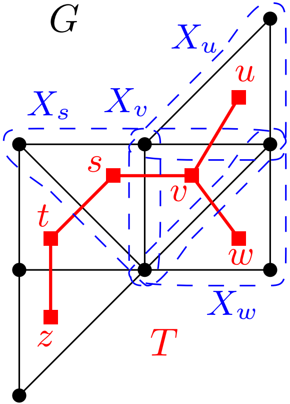
- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \ \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.
- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

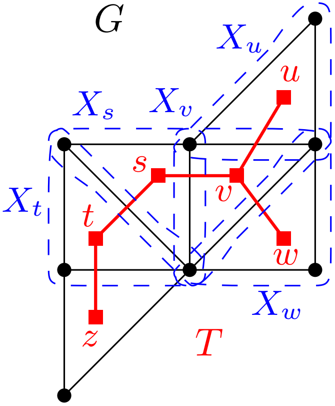
  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

- Width of a tree decomposition:

  $\max_{t \in V(T)} |X_t| - 1$.

- Treewidth of a graph G, $\text{tw}(G)$: minimum width of a tree decomposition of G.



An equivalent (and more common) definition of treewidth

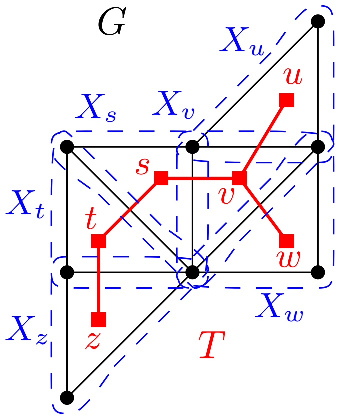
- Tree decomposition of a graph G:

  pair $(T, \{X_t \mid t \in V(T)\})$, where
  T is a tree, and
  $X_t \subseteq V(G) \;\; \forall t \in V(T)$ (bags),

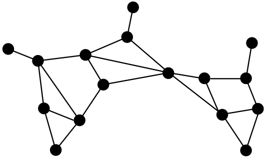
  satisfying the following:

  - $\bigcup_{t \in V(T)} X_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq X_t$.
  - $\forall v \in V(G)$, bags containing v define a connected subtree of T.

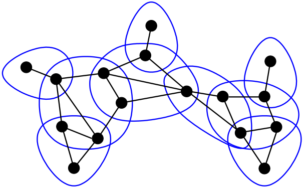
- Width of a tree decomposition:
  $\max_{t \in V(T)} |X_t| - 1$.
- Treewidth of a graph G, $\mathrm{tw}(G)$: minimum width of a tree decomposition of G.



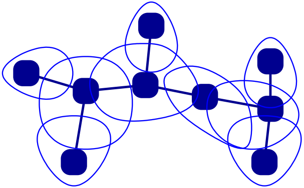
Treewidth measures the tree-likeness of a graph



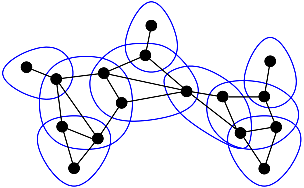
Treewidth measures the tree-likeness of a graph



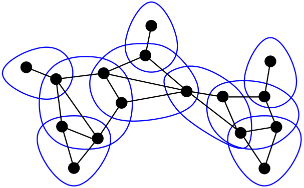
Treewidth measures the tree-likeness of a graph

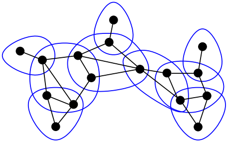


Treewidth measures the tree-likeness of a graph

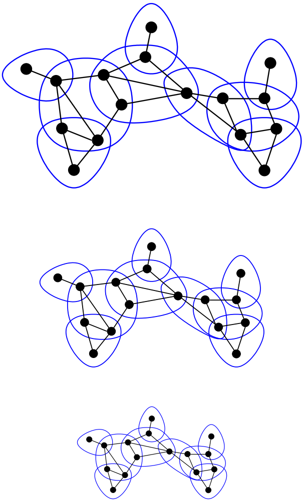


Treewidth measures the tree-likeness of a graph

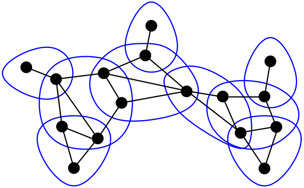


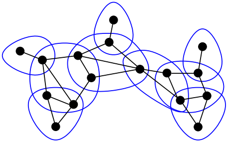


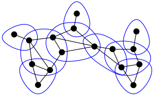
Treewidth measures the tree-likeness of a graph

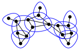


Treewidth measures the tree-likeness of a graph









Courcelle's theorem

(Credit also goes to Arnborg, Lagergreen, and Seese.)

Courcelle's theorem

Theorem (Courcelle. 1990)

The following problem is fixed-parameter tractable:

$\mathrm{MC}(\mathsf{MSO}, \mathsf{tw})$
Input: *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
Parameter: $|\varphi| + \mathsf{tw}(G)$.
Question: $G \models \varphi$?

Courcelle's theorem

Theorem (Courcelle. 1990)

The following problem is fixed-parameter tractable:

$\mathrm{MC}(\mathsf{MSO}, \mathsf{tw})$
Input: *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
Parameter: $|\varphi| + \mathsf{tw}(G)$.
Question: $G \models \varphi$?

- Can be seen as an abstraction of the notion of dynamic programming.

Theorem (Courcelle. 1990)

The following problem is fixed-parameter tractable:

> $\mathrm{MC}(\mathsf{MSO}, \mathsf{tw})$
> **Input:** *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
> **Parameter:** $|\varphi| + \mathsf{tw}(G)$.
> **Question:** $G \models \varphi$?

- Can be seen as an abstraction of the notion of dynamic programming.
- Also applies to the extension of MSO by modular counting: CMSO.

[Courcelle. 1990]

Courcelle's theorem and its extensions

Theorem (Courcelle. 1990)

The following problem is fixed-parameter tractable:

$\mathrm{MC}(\mathsf{MSO}, \mathsf{tw})$
Input: *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
Parameter: $|\varphi| + \mathsf{tw}(G)$.
Question: $G \models \varphi$?

- Can be seen as an abstraction of the notion of dynamic programming.
- Also applies to the extension of MSO by modular counting: CMSO. [Courcelle. 1990]
- Can be generalized to optimize a linear function of free second-order variables. [Arnborg, Lagergreen, Seese. 1991]

Interpretations (transductions)

One of the proofs of Courcelle's theorem uses interpretations:

Interpretations (transductions)

One of the proofs of Courcelle's theorem uses interpretations:

Class $\mathcal{D}$ | Class $\mathcal{C}$

$$G \xrightarrow{\text{algorithmic encoding}} G'$$

$$\varphi \in \text{MSO}[\tau] \xrightarrow{\text{interpretation}} \Gamma(\varphi) \in \text{MSO}[\sigma]$$

$$\Gamma(G') \cong G \xleftarrow{\text{interpretation}} G'$$

Suppose that $\mathrm{MC}($MSO$, \mathcal{C})$ is FPT

Interpretations (transductions)

One of the proofs of Courcelle's theorem uses interpretations:

Class $\mathcal{D}$		Class $\mathcal{C}$
G	$\xrightarrow{\text{algorithmic encoding}}$	G'
$\varphi \in \text{MSO}[\tau]$	$\xrightarrow{\text{interpretation}}$	$\Gamma(\varphi) \in \text{MSO}[\sigma]$
$\Gamma(G') \cong G$	$\xleftarrow{\text{interpretation}}$	G'

Suppose that $\mathrm{MC}(\text{MSO}, \mathcal{C})$ is FPT $\Rightarrow$ then $\mathrm{MC}(\text{MSO}, \mathcal{D})$ is also FPT

Interpretations (transductions)

One of the proofs of Courcelle's theorem uses interpretations:

Class $\mathcal{D}$ Class $\mathcal{C}$

$$G \xrightarrow{\text{algorithmic encoding}} G'$$

$$\varphi \in \mathrm{MSO}[\tau] \xrightarrow{\text{interpretation}} \Gamma(\varphi) \in \mathrm{MSO}[\sigma]$$

$$\Gamma(G') \cong G \xleftarrow{\text{interpretation}} G'$$

Suppose that $\mathrm{MC}(\text{MSO}, \mathcal{C})$ is FPT $\Rightarrow$ then $\mathrm{MC}(\text{MSO}, \mathcal{D})$ is also FPT

For Courcelle's theorem: interpret "tw $\leq k$" into the class of labeled trees.

Is treewidth the limit of tractability of MSO?

Is treewidth the limit of tractability of MSO? Cliquewidth!

Another crucial parameter: cliquewidth. [Courcelle, Olariu. 2000]

Is treewidth the limit of tractability of MSO?

Another crucial parameter: cliquewidth. [Courcelle, Olariu. 2000]

A graph G has cliquewidth at most k if it can be obtained by the following operations, for $i, j \in [k]$, with $i \neq j$:

- $\text{int}(v, i)$: introduce a new vertex v with color i.
- $\rho_{i \to j}$: recolor vertices of color i to color j.
- $\eta_{i,j}$: add all edges between vertices colored i and j.
- $\oplus$: take the disjoint union of two colored graphs.

Is treewidth the limit of tractability of MSO?

Another crucial parameter: cliquewidth. [Courcelle, Olariu. 2000]

A graph G has cliquewidth at most k if it can be obtained by the following operations, for $i, j \in [k]$, with $i \neq j$:

- $\text{int}(v, i)$: introduce a new vertex v with color i.
- $\rho_{i \to j}$: recolor vertices of color i to color j.
- $\eta_{i,j}$: add all edges between vertices colored i and j.
- $\oplus$: take the disjoint union of two colored graphs.

- Cliques have cliquewidth 2.
- Trees have cliquewidth 3.

Is treewidth the limit of tractability of MSO?

Another crucial parameter: cliquewidth. [Courcelle, Olariu. 2000]

A graph G has cliquewidth at most k if it can be obtained by the following operations, for $i, j \in [k]$, with $i \neq j$:

- $\text{int}(v, i)$: introduce a new vertex v with color i.
- $\rho_{i \to j}$: recolor vertices of color i to color j.
- $\eta_{i,j}$: add all edges between vertices colored i and j.
- $\oplus$: take the disjoint union of two colored graphs.

- Cliques have cliquewidth 2.
- Trees have cliquewidth 3.

Small cliquewidth does not mean "being tree-like" (such as small treewidth), but having a structure with a "tree-like decomposition".

MSO is tractable on graphs of bounded cliquewidth

- Cliques have cliquewidth 2 (but unbounded treewidth).
- Trees have cliquewidth 3.

MSO is tractable on graphs of bounded cliquewidth

- Cliques have cliquewidth 2 (but unbounded treewidth).
- Trees have cliquewidth 3.

Every graph of treewidth at most k has cliquewidth at most $2^{k+1}+1$.

[Wanke. 94 + Courcelle, Olariu. 00]

MSO is tractable on graphs of bounded cliquewidth

- Cliques have cliquewidth 2 (but unbounded treewidth).
- Trees have cliquewidth 3.

Every graph of treewidth at most k has cliquewidth at most $2^{k+1}+1$.

[Wanke. 94 + Courcelle, Olariu. 00]

Theorem (Courcelle, Makowski, Rotics. 1990)

The following problem is fixed-parameter tractable:

$\mathrm{MC}(\mathsf{MSO}, \mathsf{cw})$
Input: *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
Parameter: $|\varphi| + \mathsf{cw}(G)$.
Question: $G \models \varphi$?

MSO is tractable on graphs of bounded cliquewidth

- Cliques have cliquewidth 2 (but unbounded treewidth).
- Trees have cliquewidth 3.

Every graph of treewidth at most k has cliquewidth at most $2^{k+1} + 1$.

[Wanke. 94 + Courcelle, Olariu. 00]

Theorem (Courcelle, Makowski, Rotics. 1990)

The following problem is fixed-parameter tractable:

> $\mathrm{MC}(\mathsf{MSO}, \mathsf{cw})$
> **Input:** *A graph G and a sentence $\varphi \in \mathsf{MSO}$.*
> **Parameter:** $|\varphi| + \mathsf{cw}(G)$.
> **Question:** $G \models \varphi$?

Is the above theorem strictly more general than Courcelle's theorem?

MSO_1 and MSO_2

MSO in graphs: we allow quantification on sets of vertices.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.
2. Incidence encoding: universe $= V(G) \cup E(G)$, with the unary "vertex" and "edge" relations, and a binary "incidence" relation.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.
2. Incidence encoding: universe $= V(G) \cup E(G)$, with the unary "vertex" and "edge" relations, and a binary "incidence" relation.

Edge subdivisions preserve treewidth:

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.
2. Incidence encoding: universe $= V(G) \cup E(G)$, with the unary "vertex" and "edge" relations, and a binary "incidence" relation.

Edge subdivisions preserve treewidth:

▷ Courcelle's theorem directly generalizes to MSO_2.

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.
2. Incidence encoding: universe $= V(G) \cup E(G)$, with the unary "vertex" and "edge" relations, and a binary "incidence" relation.

Edge subdivisions preserve treewidth:

▷ Courcelle's theorem directly generalizes to MSO_2.

Edge subdivisions do not preserve cliquewidth:

MSO_1 and MSO_2

MSO_1 in graphs: we allow quantification on sets of vertices.

MSO_2 in graphs: we allow quantification on sets of vertices and edges.

Two typical ways to encode a graph G:

1. Standard encoding: universe $= V(G)$, with the binary "edge" relation.
2. Incidence encoding: universe $= V(G) \cup E(G)$, with the unary "vertex" and "edge" relations, and a binary "incidence" relation.

Edge subdivisions preserve treewidth:

- ▷ Courcelle's theorem directly generalizes to MSO_2.

Edge subdivisions do not preserve cliquewidth:

- ▷ Is it possible that $\mathrm{MC}(\mathsf{MSO}_2, \mathsf{cw})$ is FPT?

The limits of cliquewidth

EDGE DOMINATING SET, HAMILTONIAN CYCLE, and GRAPH COLORING are MSO_2-definable and W[1]-hard parameterized by cliquewidth. [Fomin, Golovach, Lokshtanov, Saurabh. 2010]

The limits of cliquewidth and MSO

Edge Dominating Set, Hamiltonian Cycle, and Graph Coloring are MSO_2-definable and W[1]-hard parameterized by cliquewidth. [Fomin, Golovach, Lokshtanov, Saurabh. 2010]

For MSO we cannot go really further than bounded treewidth/cliquewidth:

3-Colorability is NP-complete on planar graphs of degree at most 4. [Garey, Johnson, Stockmeyer. 1974]

Next section is...

FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A graph G and a sentence $\varphi \in \mathsf{FO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

As we said, this problem is in XP: solvable in time $|G|^{f(|\varphi|)}$.

FO model-checking

$\mathrm{MC}(\mathsf{FO})$
Input: A graph G and a sentence $\varphi \in \mathsf{FO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

As we said, this problem is in XP: solvable in time $|G|^{f(|\varphi|)}$.

$\mathrm{MC}(\mathsf{FO}, \mathcal{C})$
Input: A graph $G \in \mathcal{C}$ and a sentence $\varphi \in \mathsf{FO}$.
Parameter: $|\varphi|$.
Question: $G \models \varphi$?

Question: for which (parameterized) graph classes $\mathcal{C}$ is $\mathrm{MC}(\mathsf{FO}, \mathcal{C})$ FPT?

Crucial property of FO: locality

Crucial property of FO: locality

▷ A first-order formula $\varphi(x)$ on graphs is r-local if, for every graph G and every $v \in V(G)$,

$$G \models \varphi(v) \;\Leftrightarrow\; G[N_r[v]] \models \varphi],$$

where $N_r[v]$ denotes the set of vertices at distance at most r from v in G.

Crucial property of FO: locality

▷ A first-order formula $\varphi(x)$ on graphs is r-local if, for every graph G and every $v \in V(G)$,

$$G \models \varphi(v) \Leftrightarrow G[N_r[v]] \models \varphi],$$

where $N_r[v]$ denotes the set of vertices at distance at most r from v in G.

▷ A first-order formula $\varphi(x)$ on graphs is local if it is r-local for an $r \in \mathbb{N}$.

Crucial property of FO: locality

▷ A first-order formula $\varphi(x)$ on graphs is r-local if, for every graph G and every $v \in V(G)$,

$$G \models \varphi(v) \;\Leftrightarrow\; G[N_r[v]] \models \varphi],$$

where $N_r[v]$ denotes the set of vertices at distance at most r from v in G.

▷ A first-order formula $\varphi(x)$ on graphs is local if it is r-local for an $r \in \mathbb{N}$.

▷ A basic local sentence is a first-order sentence of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} \mathrm{dist}(x_i, x_j) > 2r \;\wedge\; \bigwedge_{i=1}^{k} \psi(x_i) \right),$$

where $\psi(x_i)$ is a local first-order formula.

Gaifman's theorem

Theorem (Gaifman. 1982)

Every first-order sentence is equivalent to a Boolean combination of basic local sentences, which can be effectively computed given the sentence.

Gaifman's theorem

Theorem (Gaifman. 1982)

Every first-order sentence is equivalent to a Boolean combination of basic local sentences, which can be effectively computed given the sentence.

This translation may involve a non-elementary blow-up in the size of the sentence.

[Dawar, Grohe, Kreutzer, Schweikardt. 2007]

Example: k-Dominating Set

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

Example: k-Dominating Set

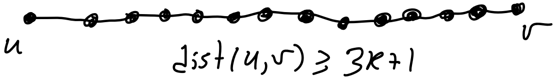
$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

To convert it into "Gaifman normal form": if diameter $\geq 3k+1$ $\rightarrow$ 'no'.

Example: k-DOMINATING SET

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

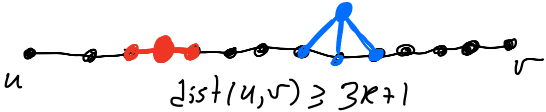
To convert it into "Gaifman normal form": if diameter $\geq 3k + 1 \rightarrow$ 'no'.



Example: k-Dominating Set

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

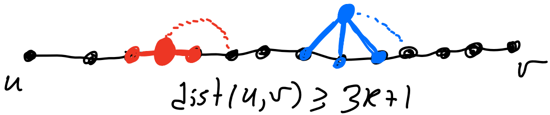
To convert it into "Gaifman normal form": if diameter $\geq 3k+1$ $\rightarrow$ 'no'.



Example: k-Dominating Set

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

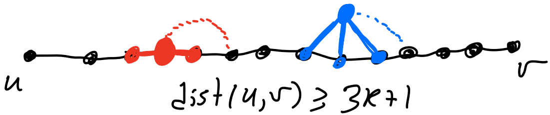
To convert it into "Gaifman normal form": if diameter $\geq 3k+1$ $\rightarrow$ 'no'.



Example: k-Dominating Set

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

To convert it into "Gaifman normal form": if diameter $\geq 3k+1$ $\rightarrow$ 'no'.



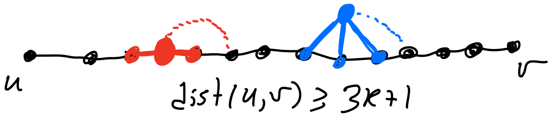
φ_k is equivalent to the conjunction of these two basic local sentences:

1. Diameter at most $3k$: $\psi := \neg\exists x_1 \exists x_2 \mathrm{dist}(x_1, x_2) \geq 3k+1$.

Example: k-Dominating Set

$$\varphi_k := \exists x_1 \ldots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee Eyx_i) \right)$$

To convert it into "Gaifman normal form": if diameter $\geq 3k + 1 \rightarrow$ 'no'.



φ_k is equivalent to the conjunction of these two basic local sentences:

1. Diameter at most $3k$: $\psi := \neg\exists x_1 \exists x_2 \text{dist}(x_1, x_2) \geq 3k + 1$.
2. $\exists x \chi(x)$, where $\chi(x)$ is the $(3k + 1)$-local formula

$$\exists y_1 \in N_{3k+1}(x) \ldots \exists y_k \in N_{3k+1}(x) \forall z \in N_{3k+1}(x) \left(\bigvee_{i=1}^{k} (y_i = z \vee Ezy_i) \right)$$

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

- Given $G \in \mathcal{C}_d$ and $\varphi \in \mathsf{FO}$, convert φ into "Gaifman normal form".

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d.
Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

- Given $G \in \mathcal{C}_d$ and $\varphi \in \mathsf{FO}$, convert φ into "Gaifman normal form".
- We only need to consider basic local sentences of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \le i < j \le k} \mathrm{dist}(x_i, x_j) > 2r \quad \wedge \quad \bigwedge_{i=1}^{k} \psi(x_i) \right).$$

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

- Given $G \in \mathcal{C}_d$ and $\varphi \in$ FO, convert φ into "Gaifman normal form".
- We only need to consider basic local sentences of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} \mathrm{dist}(x_i, x_j) > 2r \quad \wedge \quad \bigwedge_{i=1}^{k} \psi(x_i) \right).$$

- For every $v \in V(G)$, we test whether $G[N_r[v]] \models \psi$.

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathrm{FO}, \mathcal{C}_d)$ is FPT.

Proof:

- Given $G \in \mathcal{C}_d$ and $\varphi \in \mathrm{FO}$, convert φ into "Gaifman normal form".
- We only need to consider basic local sentences of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \le i < j \le k} \mathrm{dist}(x_i, x_j) > 2r \quad \wedge \quad \bigwedge_{i=1}^{k} \psi(x_i) \right).$$

- For every $v \in V(G)$, we test whether $G[N_r[v]] \models \psi$.
  Since $G[N_r[v]]$ has constant size, easy to do!

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

- Given $G \in \mathcal{C}_d$ and $\varphi \in \mathsf{FO}$, convert φ into "Gaifman normal form".
- We only need to consider basic local sentences of the form

$$\exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \le i < j \le k} \mathrm{dist}(x_i, x_j) > 2r \quad \wedge \quad \bigwedge_{i=1}^{k} \psi(x_i) \right).$$

- For every $v \in V(G)$, we test whether $G[N_r[v]] \models \psi$.
  Since $G[N_r[v]]$ has constant size, easy to do!
- Finally, we greedily try to find k such "good" vertices far apart. □

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d.
Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

- If $G[N_r[v]]$ has constant size, easy to do!

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d.
Then $\mathrm{MC}(\mathrm{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

- If $G[N_r[v]]$ has constant size, easy to do!
- But also if $\mathrm{tw}(G[N_r[v]])$ is bounded, by Courcelle's theorem.

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

- If $G[N_r[v]]$ has constant size, easy to do!
- But also if $\mathrm{tw}(G[N_r[v]])$ is bounded, by Courcelle's theorem.
- But also if $\mathrm{tw}(G[N_r[v]]) \leq f(r)$:

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

- If $G[N_r[v]]$ has constant size, easy to do!
- But also if $\mathrm{tw}(G[N_r[v]])$ is bounded, by Courcelle's theorem.
- But also if $\mathrm{tw}(G[N_r[v]]) \leq f(r)$: bounded local treewidth.

FO model-checking on graphs of bounded degree

Theorem (Seese. 1996)

Let $d \in \mathbb{N}$ and let $\mathcal{C}_d$ be the class of graphs of degree bounded by d. Then $\mathrm{MC}(\mathsf{FO}, \mathcal{C}_d)$ is FPT.

Proof:

Crucial issue: test whether $G[N_r[v]] \models \psi$ in FPT time.

- If $G[N_r[v]]$ has constant size, easy to do!
- But also if $\mathrm{tw}(G[N_r[v]])$ is bounded, by Courcelle's theorem.
- But also if $\mathrm{tw}(G[N_r[v]]) \leq f(r)$: bounded local treewidth.

This has triggered a lot of research in the last 20 years...

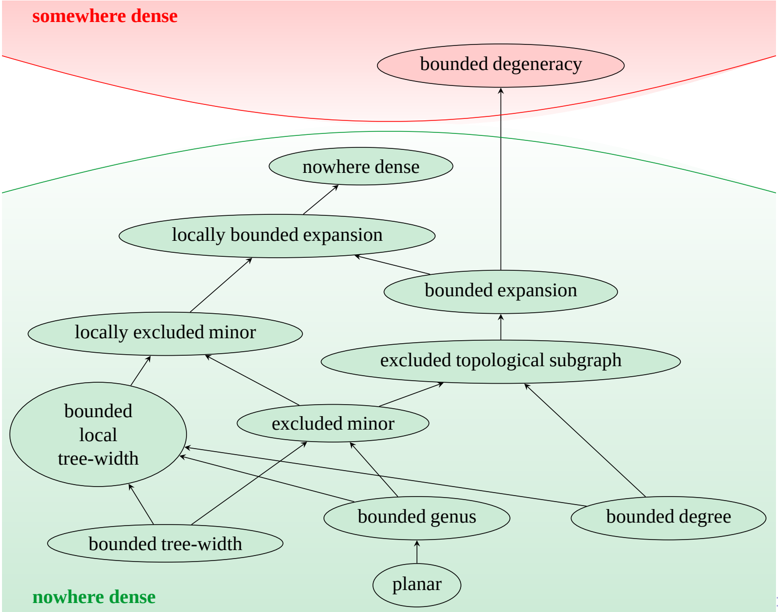
AMTs for MSO and FO

bounded treewidth [Courcelle,1990] [Arnborg, Lagergren, Seese, 1991] [Borie, Parker, Tovey, 1992]
bounded cliquewidth. [Courcelle, Makowski, Rotics, 2000] [Oum & Seymour, 2006] **MSO**

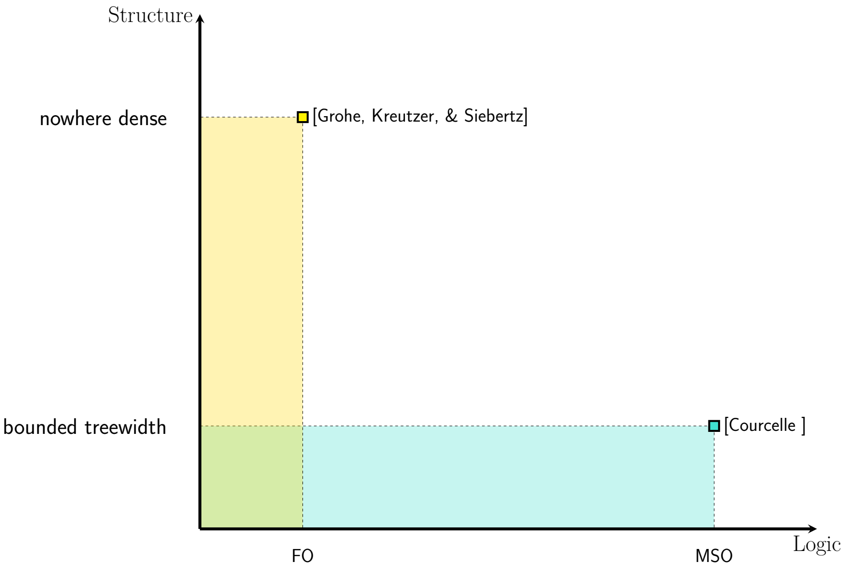
---

bounded degree [Seese, 1996] **FO**
locally bounded treewidth [Frick & Grohe, 2001]
excluding a minor [Flum & Grohe, 2001]
locally excluding a minor [Dawar, Grohe, Kreutzer, 2007]
bounded expansion [Dvořák, Kráľ, Thomas, 2011]
nowhere dense [Grohe, Kreutzer, Siebertz, 2017]
bounded twinwidth [Bonnet, Kim, Thomassé, Watrigant, 2022]
structurally bounded degree [Gajarský, Hliněný, Lokshtanov, Obdržálek, Ramanujan, 2016]
structurally bounded expansion [Gajarský, Kreutzer, Nešetřil, Ossona de Mendez, Mi. Pilipczuk, Siebertz, Toruńczyk, 2018]
structurally nowhere dense [Dreier, Mählmann, Siebertz, 2023]
structurally bounded local cliquewidth [Bonnet, Dreier, Gajarský, Kreutzer, Mählmann, Simon, Toruńczyk, 2022]
monadically stable [Dreier, Eleftheriadis, Mählmann, McCarty, Mi. Pilipczuk, Toruńczyk, 2023]
monadically NIP/dependent ?

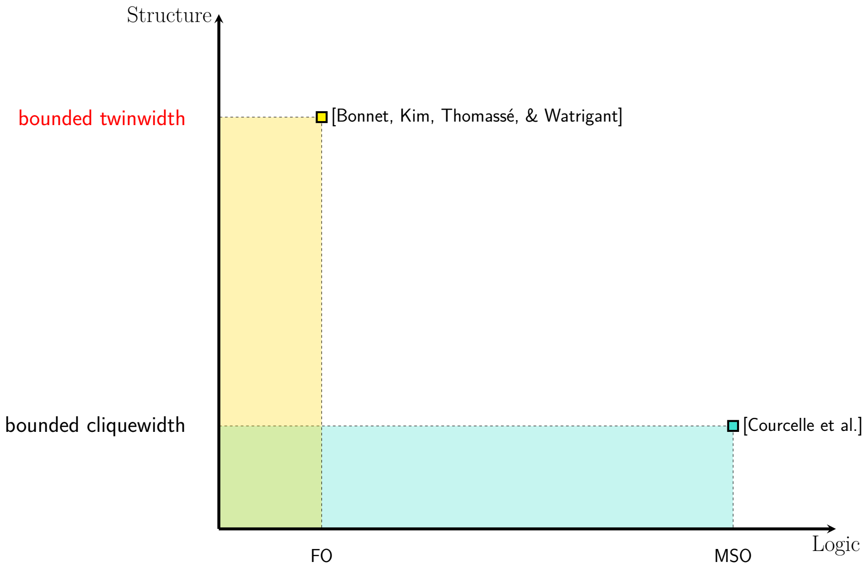
FO model-checking on sparse graph classes



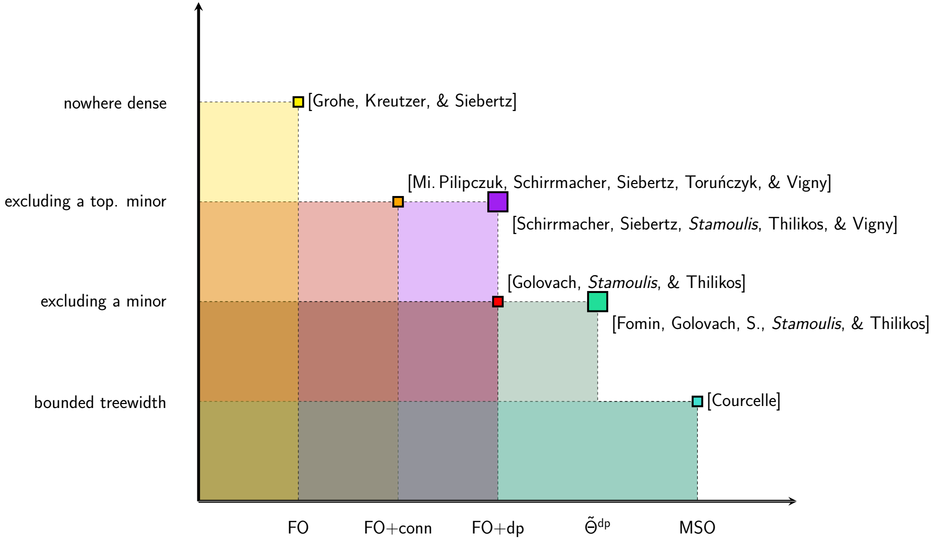
Simplified picture for monotone graph classes



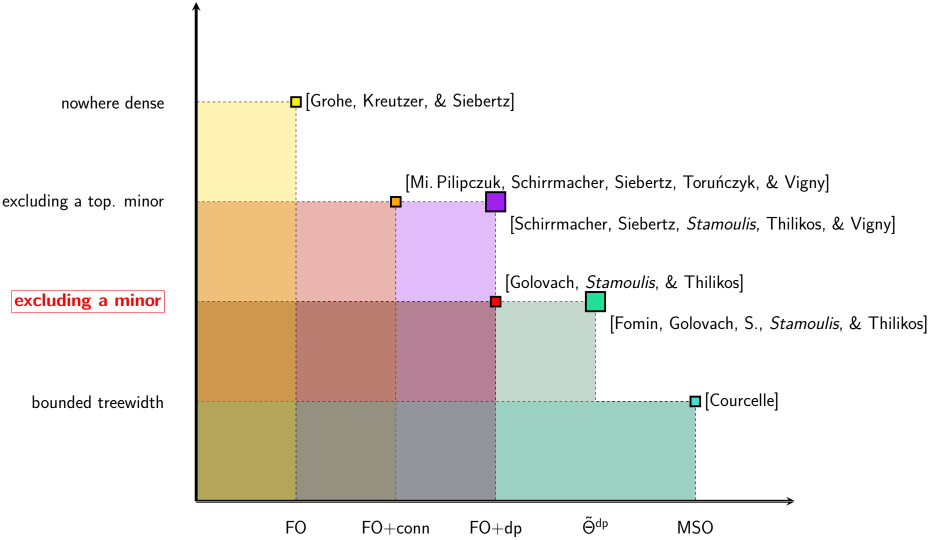
Simplified picture for hereditary graph classes



A lot of interesting stuff between FO and MSO

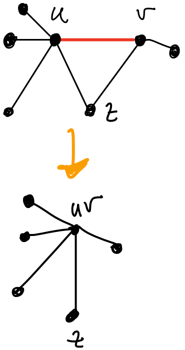


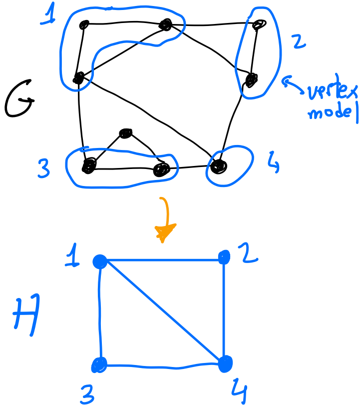
A lot of interesting stuff between FO and MSO



Graph minors

A graph H is a minor of a graph G, denoted by $H \leqslant_m G$, if H can be obtained by a subgraph of G by contracting edges.





Minor-closed graph classes

A graph class $\mathcal{C}$ is minor-closed (or closed under minors) if

$$G \in \mathcal{C} \;\Rightarrow\; H \in \mathcal{C} \text{ for every } H \leqslant_m G.$$

Minor-closed graph classes

A graph class $\mathcal{C}$ is minor-closed (or closed under minors) if

$$G \in \mathcal{C} \;\Rightarrow\; H \in \mathcal{C} \text{ for every } H \leqslant_m G.$$

Examples of minor-closed graph classes:

- Independent sets.
- Forests.
- Subgraphs of series-parallel graphs.
- Planar graphs.
- Graphs embeddable in a fixed surface.
- Linklessly embeddable graphs.
- Knotlessly embeddable graphs.
- ...

Minor-closed graph classes

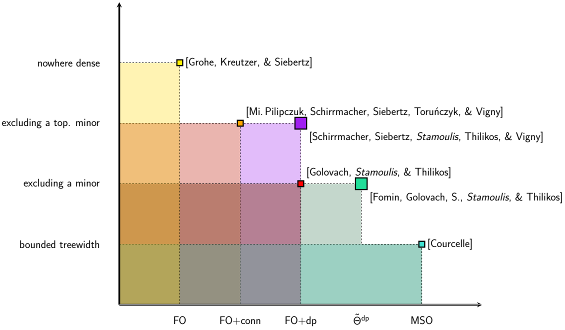
A graph class $\mathcal{C}$ is minor-closed (or closed under minors) if

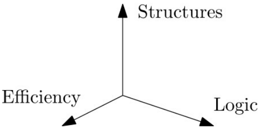
$$G \in \mathcal{C} \;\Rightarrow\; H \in \mathcal{C} \text{ for every } H \leqslant_m G.$$

Theorem (Robertson, Seymour. 1983-2004)

Every minor-closed graph class $\mathcal{C}$ can be characterized by a finite list of excluded minors.

Missing axis: efficiency dimension





$$f(|\varphi|, \mathbf{p}(G)) \cdot |G|^{\mathcal{O}(1)}$$

Gràcies!