Parameterized Algorithms

Treewidth: Courcelle's Theorem and Chordal Graphs

Thomas Bläsius

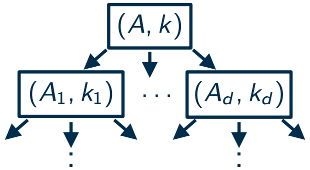
Exam dates

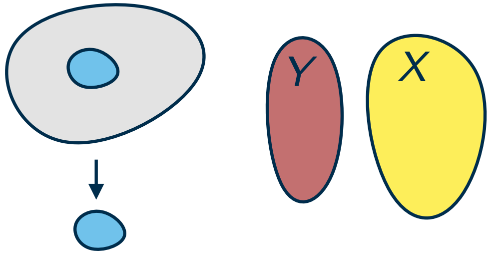
August	3				7
August	10				14
August	17				21
August	24				28
August/September	31				4
September	7				11
September	14				18
September	21				25
September	28				2

Which weeks work for you?

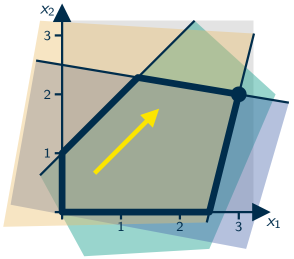
Content

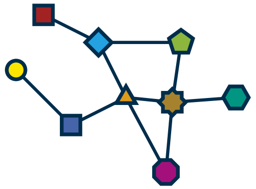
Basic toolbox
- bounded search trees
- kernelization
- iterative compression

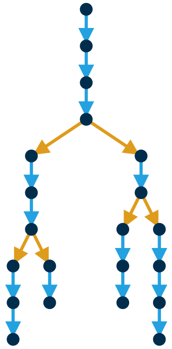




Extended toolbox
- linear programs
- branch-and-reduce
- color coding

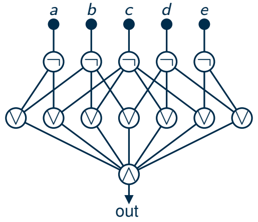


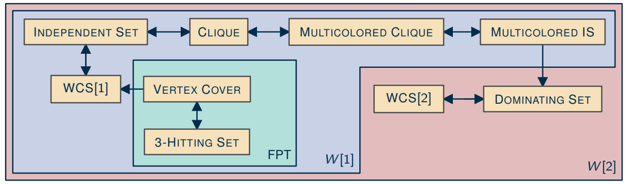




Tree width
- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

Lower bonds
- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH





Courcelle's theorem

Previously seen

- dynamic program on tree decomposition

Courcelle's theorem

Previously seen

- dynamic program on tree decomposition
- yields FPT algorithm for many problems (parameterized by treewidth):

VERTEX COVER MAXCUT STEINER TREE HAMILTON CYCLE LONGEST CYCLE CYCLE PACKING
DOMINATING SET ODD CYCLE TRANSVERSAL LONGEST PATH CONNECTED DOMINATING SET
INDEPENDENT SET FEEDBACK VERTEX SET HAMILTON PATH CONNECTED VERTEX COVER
CHROMATIC NUMBER CONNECTED FEEDBACK VERTEX SET

Courcelle's theorem

Previously seen

- dynamic program on tree decomposition
- yields FPT algorithm for many problems (parameterized by treewidth):

VERTEX COVER, MAXCUT, STEINER TREE, HAMILTON CYCLE, LONGEST CYCLE, CYCLE PACKING, DOMINATING SET, ODD CYCLE TRANSVERSAL, LONGEST PATH, CONNECTED DOMINATING SET, INDEPENDENT SET, FEEDBACK VERTEX SET, HAMILTON PATH, CONNECTED VERTEX COVER, CHROMATIC NUMBER, CONNECTED FEEDBACK VERTEX SET

Today

- meta theorem of the form:
  if a problem has property XY, then it is in FPT when parameterized by treewidth

Courcelle's theorem

Previously seen

- dynamic program on tree decomposition
- yields FPT algorithm for many problems (parameterized by treewidth):

VERTEX COVER MAXCUT STEINER TREE HAMILTON CYCLE LONGEST CYCLE CYCLE PACKING
DOMINATING SET ODD CYCLE TRANSVERSAL LONGEST PATH CONNECTED DOMINATING SET
INDEPENDENT SET FEEDBACK VERTEX SET HAMILTON PATH CONNECTED VERTEX COVER
CHROMATIC NUMBER CONNECTED FEEDBACK VERTEX SET

Today

- meta theorem of the form:
  if a problem has property XY, then it is in FPT when parameterized by treewidth
- we don't prove the theorem here

Courcelle's theorem

Previously seen

- dynamic program on tree decomposition
- yields FPT algorithm for many problems (parameterized by treewidth):

VERTEX COVER MAXCUT STEINER TREE HAMILTON CYCLE LONGEST CYCLE CYCLE PACKING
DOMINATING SET ODD CYCLE TRANSVERSAL LONGEST PATH CONNECTED DOMINATING SET
INDEPENDENT SET FEEDBACK VERTEX SET HAMILTON PATH CONNECTED VERTEX COVER
CHROMATIC NUMBER CONNECTED FEEDBACK VERTEX SET

Today

- meta theorem of the form:
  if a problem has property XY, then it is in FPT when parameterized by treewidth
- we don't prove the theorem here
- but I'll tell you, what XY is

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\quad\Leftrightarrow\quad$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in X}\, u \in Y \wedge v \notin Y) \\ \Rightarrow (\exists_{e \in E} \exists_{u,v \in X}\, \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \Big]$$

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\Leftrightarrow$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \\ \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y)\big]$$

- solution: $A(X) = \text{true}$ $\Leftrightarrow$ $X \subseteq V$ induces a connected graph

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\Leftrightarrow$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X}\, u \in Y \land v \notin Y) \\ \Rightarrow (\exists_{e \in E} \exists_{u,v \in X}\, \text{inc}(u, e) \land \text{inc}(v, e) \land u \in Y \land v \notin Y) \big]$$

- solution: $A(X) = \text{true} \Leftrightarrow X \subseteq V$ induces a connected graph

What is allowed in MSO_2?

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\Leftrightarrow$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in X}\, u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X}\, \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \Big]$$

- solution: $A(X) = \text{true}$ $\Leftrightarrow$ $X \subseteq V$ induces a connected graph

What is allowed in MSO_2?

Second Order: you can quantify over elements (first order) and over relations (second order)

first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ $\forall_{Y \subseteq V \times V}$ $\forall_{Y \subseteq V \times V \times V}$

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\Leftrightarrow$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X} u \in Y \land v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \land \text{inc}(v, e) \land u \in Y \land v \notin Y) \big]$$

- solution: $A(X) = \text{true}$ $\Leftrightarrow$ $X \subseteq V$ induces a connected graph

What is allowed in MSO_2?

Second Order: you can quantify over elements (first order) and over relations (second order)

Monadic: not all relations, only sets

first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ ~~$\forall_{Y \subseteq V \times V}$~~ ~~$\forall_{Y \subseteq V \times V \times V}$~~

MSO_2 on graphs

(MSO = monadic second order logic)

Example: What is the meaning of the following formula?

- $G = (V, E)$ is a graph and $X \subseteq V$
- for $v \in V$ and $e \in E$, $\text{inc}(v, e)$ is true $\Leftrightarrow$ v is a vertex of e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X}\, u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X}\, \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \big]$$

- solution: $A(X) = \text{true} \Leftrightarrow X \subseteq V$ induces a connected graph

What is allowed in MSO_2?

Second Order: you can quantify over elements (first order) and over relations (second order)

Monadic: not all relations, only sets

2: you can quantify over V and over E
(MSO₁: only over V)

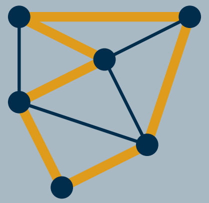
first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ ~~$\forall_{Y \subseteq V \times V}$~~ ~~$\forall_{Y \subseteq V \times V \times V}$~~

$\forall_{v \in V}$ $\forall_{e \in E}$ $\forall_{V' \subseteq V}$ $\forall_{E' \subseteq E}$

Formulating Hamiltonian Cycle in MSO_2

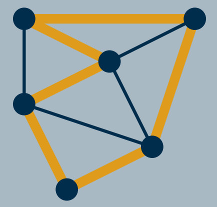
Problem: Hamiltonian Cycle
Given a graph G. Does G have a cycle that visits all vertices?



Express existence of a Hamiltonian cycle in MSO_2

Formulating HAMILTONIAN CYCLE in MSO_2

Problem: HAMILTONIAN CYCLE
Given a graph G. Does G have a cycle that visits all vertices?

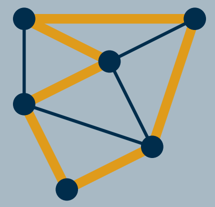


Express existence of a Hamiltonian cycle in MSO_2

- note: $C \subseteq E$ is solution $\Leftrightarrow$ $G[C]$ is connected and 2-regular

Formulating Hamiltonian Cycle in MSO_2

Problem: Hamiltonian Cycle
Given a graph G. Does G have a cycle that visits all vertices?

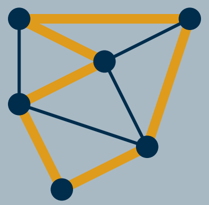


Express existence of a Hamiltonian cycle in MSO_2

- note: $C \subseteq E$ is solution $\Leftrightarrow$ $G[C]$ is connected and 2-regular
- high-level formula: $\text{hamiltonian} = \exists_{C \subseteq E}\, \text{conn}(C) \wedge \forall_{v \in V}\, \text{deg2}(v, C)$

Formulating HAMILTONIAN CYCLE in MSO_2

Problem: HAMILTONIAN CYCLE
Given a graph G. Does G have a cycle that visits all vertices?



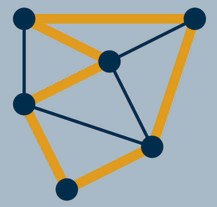
Express existence of a Hamiltonian cycle in MSO_2

- note: $C \subseteq E$ is solution $\Leftrightarrow$ $G[C]$ is connected and 2-regular
- high-level formula: $\text{hamiltonian} = \exists_{C \subseteq E}\, \text{conn}(C) \wedge \forall_{v \in V}\, \text{deg2}(v, C)$
- connectedness:

$$\text{conn}(C) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in V}\, u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in C} \exists_{u \in Y} \exists_{v \notin Y}\, \text{inc}(u, e) \wedge \text{inc}(v, e) \Big]$$

Formulating HAMILTONIAN CYCLE in MSO_2

Problem: HAMILTONIAN CYCLE
Given a graph G. Does G have a cycle that visits all vertices?



Express existence of a Hamiltonian cycle in MSO_2

- note: $C \subseteq E$ is solution $\Leftrightarrow$ $G[C]$ is connected and 2-regular
- high-level formula: $$\text{hamiltonian} = \exists_{C \subseteq E}\, \text{conn}(C) \wedge \forall_{v \in V}\, \text{deg2}(v, C)$$
- connectedness: $$\text{conn}(C) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in V}\, u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in C} \exists_{u \in Y} \exists_{v \notin Y}\, \text{inc}(u, e) \wedge \text{inc}(v, e)) \Big]$$
- 2-regularity: $$\text{deg2}(v, C) = \exists_{e_1, e_2 \in C} \Big[e_1 \neq e_2 \wedge \text{inc}(v, e_1) \wedge \text{inc}(v, e_2) \wedge (\forall_{e_3 \in C}\, \text{inc}(v, e_3) \Rightarrow (e_3 = e_1 \vee e_3 = e_2)) \Big]$$

Courcelle's theorem

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Courcelle's theorem

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Example

- property to have a Hamiltonian cycle: can be expressed with a MSO_2 formula of constant size
- thus: HAMILTONIAN CYCLE parameterized with treewidth is FPT

Courcelle's theorem

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Example

- property to have a Hamiltonian cycle: can be expressed with a MSO_2 formula of constant size
- thus: HAMILTONIAN CYCLE parameterized with treewidth is FPT

What are the free variables for?

- useful, if the input is not just the graph but, e.g., consists of $G = (V, E)$ and $X \subseteq V$

Courcelle's theorem

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Example

- property to have a Hamiltonian cycle: can be expressed with a MSO_2 formula of constant size
- thus: HAMILTONIAN CYCLE parameterized with treewidth is FPT

What are the free variables for?

- useful, if the input is not just the graph but, e.g., consists of $G = (V, E)$ and $X \subseteq V$
- example: for STEINER TREE, the input consists of a graph and a set of terminal vertices

Courcelle's theorem – another example

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Another example: k-VERTEX COVER

Courcelle's theorem – another example

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Another example: *k*-Vertex Cover

- high-level formula: $\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$

Courcelle's theorem – another example

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Another example: *k*-Vertex Cover

- high-level formula: $\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$
- covering all edges: $\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$

Courcelle's theorem – another example

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Another example: *k*-VERTEX COVER

- high-level formula: $\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$
- covering all edges: $\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \, \text{inc}(v, e)$
- at most k vertices: $\text{size-k}(X) = \forall_{v_0, \ldots, v_k \in V} \, v_0 \notin X \vee \cdots \vee v_k \notin X \vee v_0 = v_1 \vee v_0 = v_2 \vee \cdots \vee v_{k-1} = v_k$

  (for every set of $k + 1$ vertices, one is not in X or two are the same vertex)

Courcelle's theorem – another example

Theorem (without proof)
Let φ be a MSO_2 formula and let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables in φ. Then there is an algorithm, that checks in $f(|\varphi|, t) \cdot n$ time whether G satisfies φ (for a computable function f).

Another example: *k*-VERTEX COVER

- high-level formula: $\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$
- covering all edges: $\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$
- at most k vertices: $\text{size-k}(X) = \forall_{v_0, \dots, v_k \in V} v_0 \notin X \vee \dots \vee v_k \notin X \vee v_0 = v_1 \vee v_0 = v_2 \vee \dots \vee v_{k-1} = v_k$
  (for every set of $k+1$ vertices, one is not in X or two are the same vertex)

Note: $|\varphi| = |\text{k-vc}|$ depends on k
$\Rightarrow$ Courcelle's theorem only gives an FPT-algorithm for parameter $k + t$

Courcelle's theorem – optimization

Theorem (without proof)
Let φ be a MSO_2 formula with p free monadic variables $X_1, \dots, X_p$ and let $\alpha(x_1, \dots, x_p)$ be an affine function. Let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables of φ except $X_1, \dots, X_p$. Then there is an algorithm that finds in $f(|\varphi|, t) \cdot n$ time values for $X_1, \dots X_p$, such that $\varphi(X_1, \dots, X_p) = \texttt{true}$ and $\alpha(|X_1|, \dots, |X_p|)$ is minimum or maximum.

affine function: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelle's theorem – optimization

Theorem (without proof)
Let φ be a MSO_2 formula with p free monadic variables $X_1, \dots, X_p$ and let $\alpha(x_1, \dots, x_p)$ be an affine function. Let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables of φ except $X_1, \dots, X_p$. Then there is an algorithm that finds in $f(|\varphi|, t) \cdot n$ time values for $X_1, \dots X_p$, such that $\varphi(X_1, \dots, X_p) = \texttt{true}$ and $\alpha(|X_1|, \dots, |X_p|)$ is minimum or maximum.

Example: Vertex Cover

affine function: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelle's theorem – optimization

Theorem (without proof)
Let φ be a MSO_2 formula with p free monadic variables $X_1, \dots, X_p$ and let $\alpha(x_1, \dots, x_p)$ be an affine function. Let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables of φ except $X_1, \dots, X_p$. Then there is an algorithm that finds in $f(|\varphi|, t) \cdot n$ time values for $X_1, \dots X_p$, such that $\varphi(X_1, \dots, X_p) = \mathtt{true}$ and $\alpha(|X_1|, \dots, |X_p|)$ is minimum or maximum.

affine function: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Example: VERTEX COVER

- one free variable $X \subseteq V$
- goal 1: X is a vertex cover: $\varphi(X) = \mathrm{vc}(X) = \forall_{e \in E} \exists_{v \in X} \, \mathrm{inc}(v, e)$

Courcelle's theorem – optimization

Theorem (without proof)
Let φ be a MSO_2 formula with p free monadic variables $X_1, \dots, X_p$ and let $\alpha(x_1, \dots, x_p)$ be an affine function. Let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables of φ except $X_1, \dots, X_p$. Then there is an algorithm that finds in $f(|\varphi|, t) \cdot n$ time values for $X_1, \dots X_p$, such that $\varphi(X_1, \dots, X_p) = \mathtt{true}$ and $\alpha(|X_1|, \dots, |X_p|)$ is minimum or maximum.

affine function: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Example: VERTEX COVER

- one free variable $X \subseteq V$
- goal 1: X is a vertex cover: $\varphi(X) = \mathrm{vc}(X) = \forall_{e \in E} \exists_{v \in X} \, \mathrm{inc}(v, e)$
- goal 2: X is minimum among all vertex covers: $\alpha(x) = x$

Courcelle's theorem – optimization

Theorem (without proof)
Let φ be a MSO_2 formula with p free monadic variables $X_1, \dots, X_p$ and let $\alpha(x_1, \dots, x_p)$ be an affine function. Let G be a graph together with a tree decomposition of width t and with an evaluation of all free variables of φ except $X_1, \dots, X_p$. Then there is an algorithm that finds in $f(|\varphi|, t) \cdot n$ time values for $X_1, \dots X_p$, such that $\varphi(X_1, \dots, X_p) = \texttt{true}$ and $\alpha(|X_1|, \dots, |X_p|)$ is minimum or maximum.

affine function: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Example: Vertex Cover

- one free variable $X \subseteq V$
- goal 1: X is a vertex cover: $\varphi(X) = \mathrm{vc}(X) = \forall_{e \in E} \exists_{v \in X} \, \mathrm{inc}(v, e)$
- goal 2: X is minimum among all vertex covers: $\alpha(x) = x$
- $|\varphi|$ is constant $\Rightarrow$ Vertex Cover parameterized by treewidth is FPT

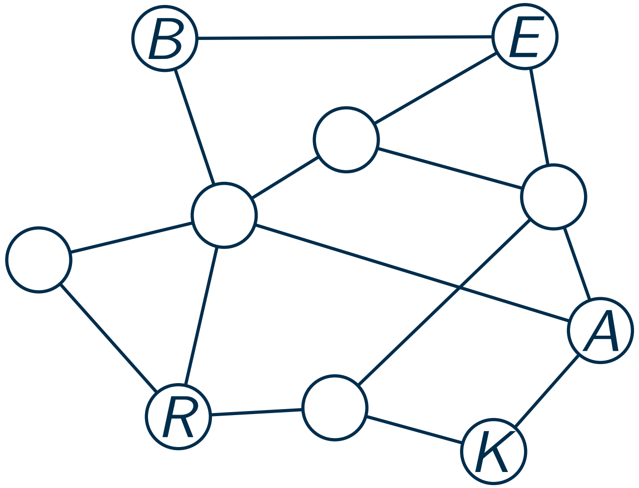
What is the problem?

Another example

- formula $\varphi(X)$ with free variables X and T

$$\varphi(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in T}\, u \in Y \land v \notin Y) \Rightarrow (\exists_{u,v \in V} \exists_{e \in X}\, u \in Y \land v \notin Y \land \mathrm{inc}(u,e) \land \mathrm{inc}(v,e)) \Big]$$

- $\alpha(x) = x$
- instance: $G = (V, E)$ is the shown graph and $T = \{B, R, E, A, K\} \subseteq V$



Your task: find $X \subseteq E$, such that $\alpha(|X|)$ is minimized

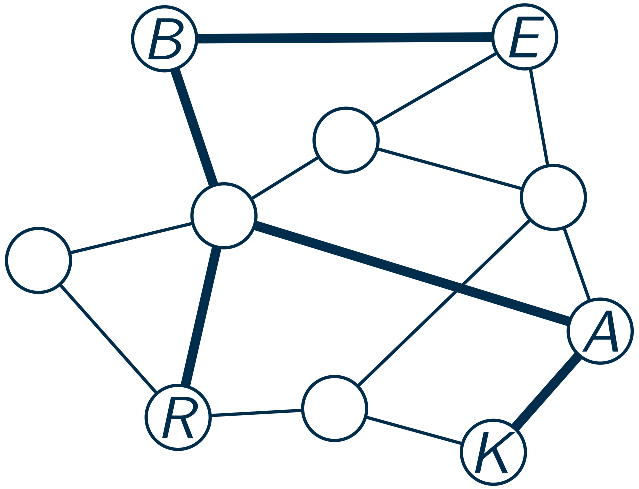
What is the problem?

Another example

- formula $\varphi(X)$ with free variables X and T

$$\varphi(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in T}\, u \in Y \wedge v \notin Y) \Rightarrow (\exists_{u,v \in V} \exists_{e \in X}\, u \in Y \wedge v \notin Y \wedge \mathrm{inc}(u,e) \wedge \mathrm{inc}(v,e)) \Big]$$

- $\alpha(x) = x$
- instance: $G = (V, E)$ is the shown graph and $T = \{B, R, E, A, K\} \subseteq V$



Your task: find $X \subseteq E$, such that $\alpha(|X|)$ is minimized

Solution: connected subgraph with minimum number of edges that contains all vertices of $T \Rightarrow$ Steiner Tree

Lecture evaluation
(the exercise sessions are evaluated separately)

1.19 What I liked most:

1.20 What I did not like at all:

Lecture evaluation

(the exercise sessions are evaluated separately)

What was good and should be kept?

1.19 ~~What I liked most:~~

How can the lecture be improved?

1.20 ~~What I did not like at all:~~

Lecture evaluation

(the exercise sessions are evaluated separately)

What was good and should be kept?

1.19 ~~What I liked most:~~

How can the lecture be improved?

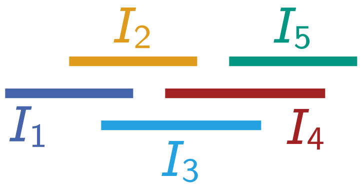
1.20 ~~What I did not like at all:~~

https://onlineumfrage.kit.edu/evasys/online.php?p=K4TXX

Interval graphs

Definition

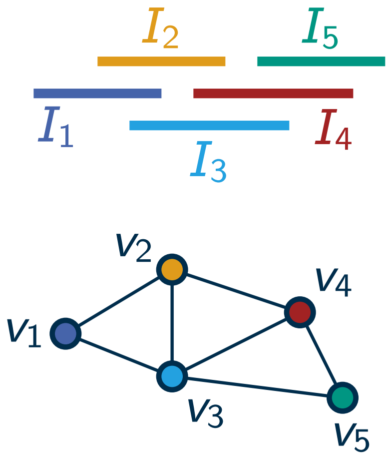
- set of n intervals $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$



Interval graphs

Definition

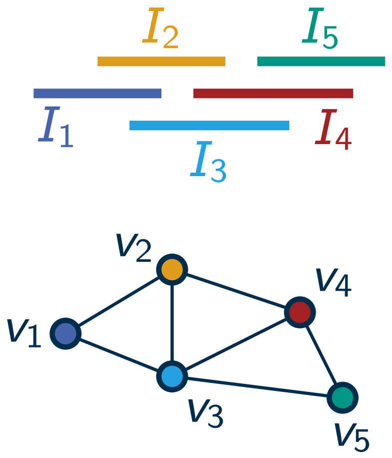
- set of n intervals $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$



Interval graphs

Definition

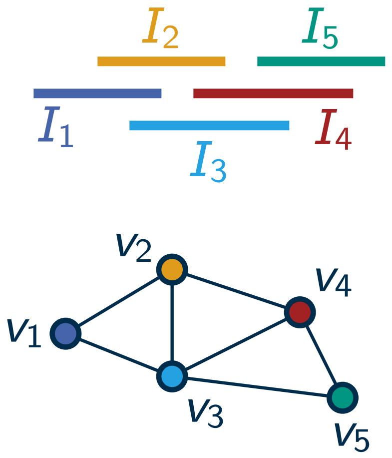
- set of n intervals $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \ldots, I_n$ is the **interval representation** of G



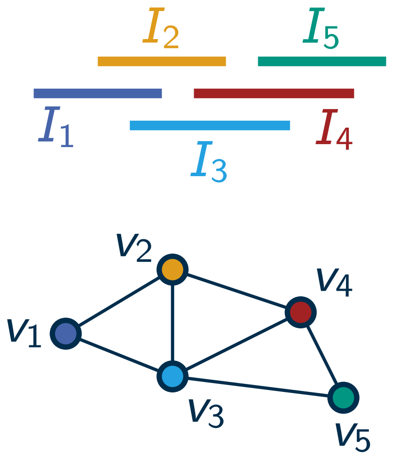
Interval graphs

Definition

- set of n intervals $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \ldots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation



Interval graphs



Definition

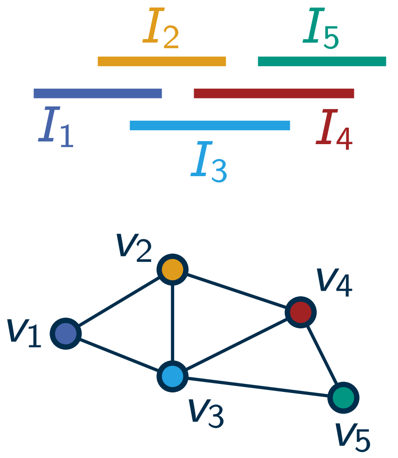
- set of n intervals $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation

What is the pathwidth of an interval graph?

Interval graphs

Definition

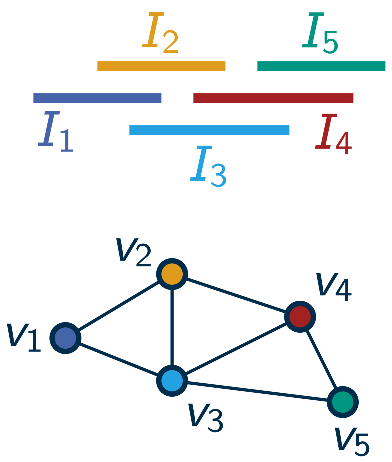
- set of n intervals $\{I_1, ..., I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, ..., I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation



What is the pathwidth of an interval graph?

- let $\omega(G)$ be the clique number and $\mathrm{pw}(G)$ be the pathwidth of G

Interval graphs



Definition

- set of n intervals $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation

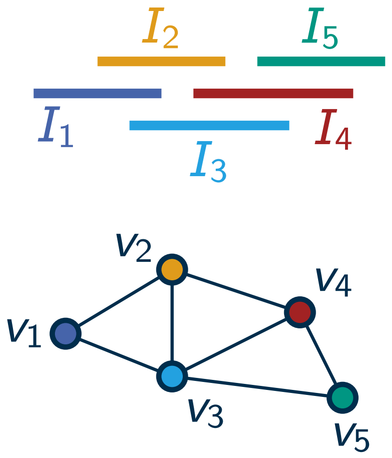
What is the pathwidth of an interval graph?

- let $\omega(G)$ be the clique number and $\mathrm{pw}(G)$ be the pathwidth of G
- then: $\mathrm{pw}(G) \leq \omega(G) - 1$ (interval representation yields path decomposition)

Interval graphs

Definition

- set of n intervals $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \ldots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation



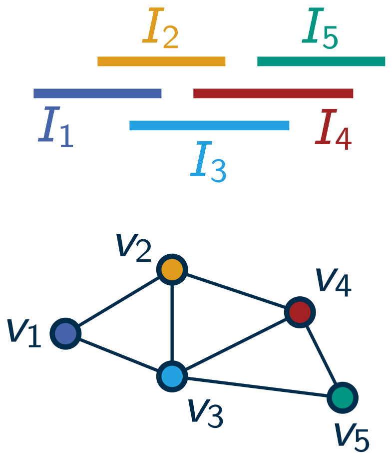
What is the pathwidth of an interval graph?

- let $\omega(G)$ be the clique number and $\text{pw}(G)$ be the pathwidth of G
- then: $\text{pw}(G) \leq \omega(G) - 1$ (interval representation yields path decomposition)
- and: $\text{pw}(G) \geq \omega(G) - 1$ (vertices of a clique must share a bag)

Interval graphs

Definition

- set of n intervals $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \ldots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation

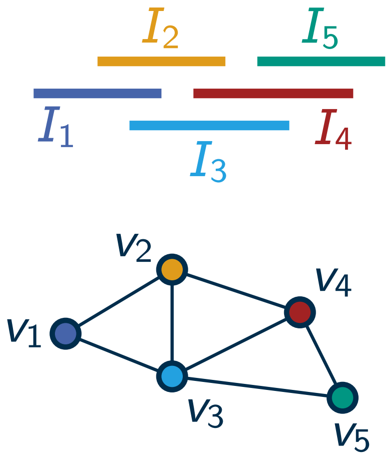


What is the pathwidth of an interval graph?

- let $\omega(G)$ be the clique number and $\text{pw}(G)$ be the pathwidth of G
- then: $\text{pw}(G) \leq \omega(G) - 1$ (interval representation yields path decomposition)
- and: $\text{pw}(G) \geq \omega(G) - 1$ (vertices of a clique must share a bag)

$\Rightarrow \text{pw}(G) = \omega(G) - 1$

Interval graphs



Definition

- set of n intervals $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ is the **interval representation** of G
- a graph is an **interval graph** $\Leftrightarrow$ it has an interval representation

What is the pathwidth of an interval graph?

- let $\omega(G)$ be the clique number and $\mathrm{pw}(G)$ be the pathwidth of G
- then: $\mathrm{pw}(G) \leq \omega(G) - 1$ (interval representation yields path decomposition)
- and: $\mathrm{pw}(G) \geq \omega(G) - 1$ (vertices of a clique must share a bag)

$\Rightarrow \mathrm{pw}(G) = \omega(G) - 1$

Interval width

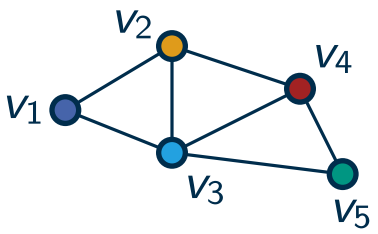
- $\text{interval-width}(G) = \min\{\omega(G') \mid G \subseteq G' \text{ and } G' \text{ is an interval graph}\}$
- $\Rightarrow \mathrm{pw}(G) = \text{interval-width}(G) - 1$

Chordal graphs

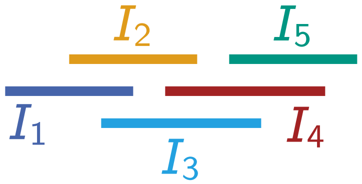
Definition (instead of subpaths of a path as for interval graphs)

A graph is **chordal** if it has an intersection representation with subtrees of a tree.

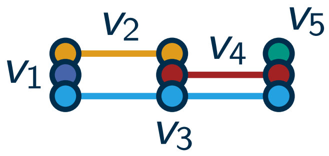
interval graph



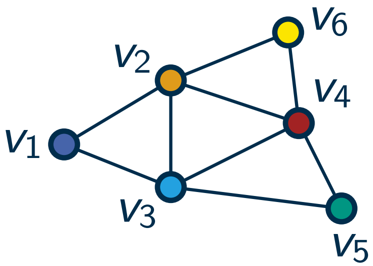
interval representation



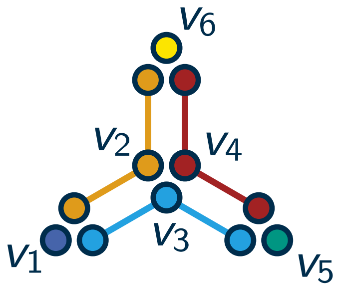
subpaths of a path



chordal graph

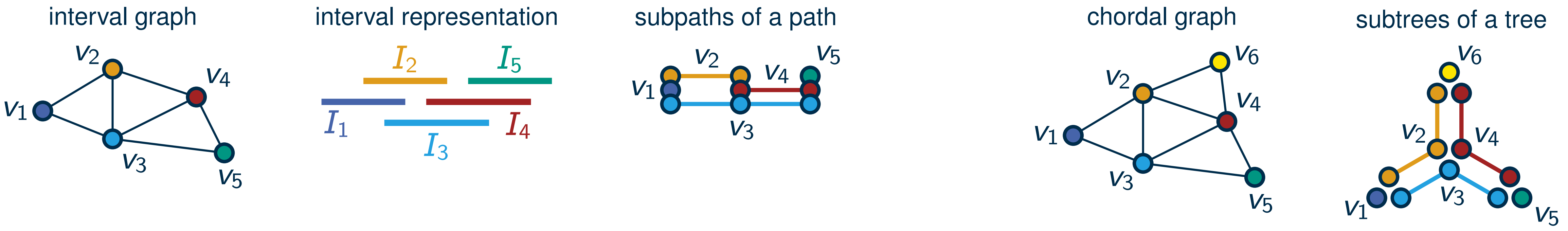


subtrees of a tree



Chordal graphs

Definition (instead of subpaths of a path as for interval graphs)
A graph is **chordal** if it has an intersection representation with subtrees of a tree.



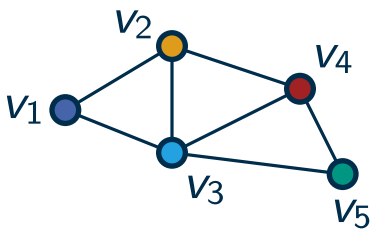
Equivalent definition

- G chordal $\Leftrightarrow$ G has no induced cycle of length at least 4

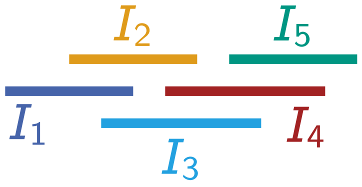
Chordal graphs

Definition (instead of subpaths of a path as for interval graphs)
A graph is **chordal** if it has an intersection representation with subtrees of a tree.

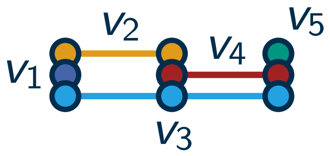
interval graph



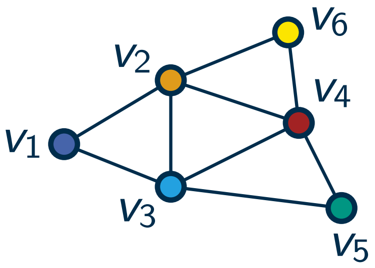
interval representation



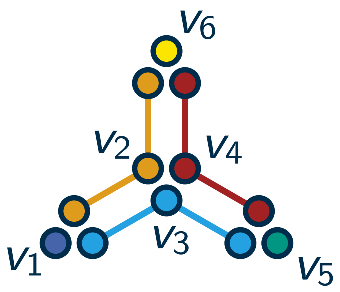
subpaths of a path



chordal graph



subtrees of a tree



Equivalent definition

- G chordal $\Leftrightarrow$ G has no induced cycle of length at least 4

Chordal width

- $\text{chordal-width}(G) = \min\{\omega(G') \mid G \subseteq G' \text{ and } G' \text{ is chordal}\}$
- $\Rightarrow \text{tw}(G) = \text{chordal-width}(G) - 1$

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2
- **Warning:** Courcelle yields really bad running times:

Wrap-Up

Courcelle's theorem

- problem can be formulated in MSO_2 $\Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2
- **Warning:** Courcelle yields really bad running times:
  - $f(k)$ is multi-exponential; think of: $2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2
- **Warning:** Courcelle yields really bad running times:
  - $f(k)$ is multi-exponential; think of: $2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$
  - height of the power tower is not bounded by a constant

(linear in the number of alternations between $\forall$ and $\exists$)

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2
- **Warning:** Courcelle yields really bad running times:
  - $f(k)$ is multi-exponential; think of: $2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$
  - height of the power tower is not bounded by a constant

(linear in the number of alternations between $\forall$ and $\exists$)

Chordal graphs

- treewidth $=$ smallest clique number of chordal supergraph

Wrap-Up

Courcelle's theorem

- problem can be formulated in $MSO_2 \Rightarrow$ FPT when parameterized by treewidth
- powerful tool, as many problems can be formulated in MSO_2
- **Warning:** Courcelle yields really bad running times:
  - $f(k)$ is multi-exponential; think of: $2^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}$
  - height of the power tower is not bounded by a constant

    (linear in the number of alternations between $\forall$ and $\exists$)

Chordal graphs

- treewidth $=$ smallest clique number of chordal supergraph
- clique number and optimal tree decomposition of chordal graphs can be computed efficiently

Literature

The monadic second-order logic of graphs. I. Recognizable sets of finite graphs

- Bruno Courcelle [1990]
- Courcelle's Theorem `https://doi.org/10.1016/0890-5401(90)90043-H`

Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families

- Richard B. Borie, R. Gary Parker, Craig A. Tovey [1992]
- independent rediscovery `https://doi.org/10.1007/BF01758777`

Courcelle's theorem—A game-theoretic approach

- Joachim Kneis, Alexander Langer, Peter Rossmanith [2011]
- Courcelle's theorem in "practice" `https://doi.org/10.1016/j.disopt.2011.06.001`