Monadic Second Order logic on graphs and Courcelle's theorem

Michał Pilipczuk

Institutt for Informatikk, Universitetet i Bergen

August 20$^{\text{th}}$, 2014

Idea

- Very roughly: a systematic way for constructing dynamic programming on tree decompositions from problem descriptions.

Idea

- Very roughly: a systematic way for constructing dynamic programming on tree decompositions from problem descriptions.
- Logical description $\rightsquigarrow$ Algorithm.

Idea

- Very roughly: a systematic way for constructing dynamic programming on tree decompositions from problem descriptions.
- Logical description $\rightsquigarrow$ Algorithm.
- Language for logical descriptions:
  MSO$_2$, Monadic Second Order logic.

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u,e) \wedge \mathbf{inc}(v,e) \\ & \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u,e) \wedge \mathbf{inc}(v,e) \\ & \quad \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u, e) \wedge \mathbf{inc}(v, e) \\ & \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u,e) \wedge \mathbf{inc}(v,e) \\ & \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u,e) \wedge \mathbf{inc}(v,e) \\ & \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

$\mathbf{MSO}_2$ on an example

- **Goal**: For a graph G and vertex subset $X \subseteq V(G)$, we want to express the property that $G[X]$ is connected.
- **English**: *For every subset of vertices Y, if X contains both a vertex from Y and outside of Y, then there exists an edge e whose endpoints u, v both belong to X, but one of them is in Y and the other is outside of Y.*
- $\mathbf{MSO}_2$:

$$\begin{aligned}\mathbf{conn}(X) = \quad & \forall_{Y \subseteq V}[(\exists_{u \in X}\, u \in Y \wedge \exists_{v \in X}\, v \notin Y) \\ & \Rightarrow (\exists_{e \in E}\, \exists_{u \in X}\, \exists_{v \in X}\, \mathbf{inc}(u,e) \wedge \mathbf{inc}(v,e) \\ & \wedge u \in Y \wedge v \notin Y)].\end{aligned}$$

Variables

- Formal syntax and semantics is in the book; here a sketch.

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;
  - $e, f, \ldots$ for edges;

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;
  - $e, f, \ldots$ for edges;
  - $X, Y, \ldots$ for subsets of vertices;

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;
  - $e, f, \ldots$ for edges;
  - $X, Y, \ldots$ for subsets of vertices;
  - $A, B, \ldots$ for subsets of edges.

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;
  - $e, f, \ldots$ for edges;
  - $X, Y, \ldots$ for subsets of vertices;
  - $A, B, \ldots$ for subsets of edges.
- Variables are *evaluated* to objects in the input graph.

Variables

- Formal syntax and semantics is in the book; here a sketch.
- Formulas of $\mathbf{MSO}_2$ have *variables*:
  - $u, v, w, \ldots$ for vertices;
  - $e, f, \ldots$ for edges;
  - $X, Y, \ldots$ for subsets of vertices;
  - $A, B, \ldots$ for subsets of edges.
- Variables are *evaluated* to objects in the input graph.
- A formula can have *free variables*; think of them as parameters.

Evaluation of a formula

- Formulas are *evaluated* in graphs, possibly enriched with evaluation of free variables.

Evaluation of a formula

- Formulas are *evaluated* in graphs, possibly enriched with evaluation of free variables.
- Thus a formula can be *true* or *false*.

Evaluation of a formula

- Formulas are *evaluated* in graphs, possibly enriched with evaluation of free variables.
- Thus a formula can be *true* or *false*.
- Denoted $\langle G, X \rangle \models \mathbf{conn}$.

Evaluation of a formula

- Formulas are *evaluated* in graphs, possibly enriched with evaluation of free variables.
- Thus a formula can be *true* or *false*.
- Denoted $\langle G, X \rangle \models \mathbf{conn}$.
- Think of an evaluation of a formula in a graph as a run of a program.

Inductive definition

- Formulas are defined inductively from simpler formulas.

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).
  - $u = v$: two objects are equal.

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).
  - $u = v$: two objects are equal.
  - $\mathbf{inc}(u, e)$: edge e is incident with vertex u.

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).
  - $u = v$: two objects are equal.
  - $\mathbf{inc}(u, e)$: edge e is incident with vertex u.
- We can use logical connectives:

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).
  - $u = v$: two objects are equal.
  - $\mathbf{inc}(u, e)$: edge e is incident with vertex u.
- We can use logical connectives:
  - Based on φ_1, φ_2, we can construct $\varphi_1 \wedge \varphi_2$, $\varphi_1 \Rightarrow \varphi_2$, etc.

Inductive definition

- Formulas are defined inductively from simpler formulas.
- Simplest building blocks: atomic formulas.
  - $u \in X$: vertex u belongs to the set X (similarly for edges).
  - $u = v$: two objects are equal.
  - $\mathbf{inc}(u, e)$: edge e is incident with vertex u.
- We can use logical connectives:
  - Based on φ_1, φ_2, we can construct $\varphi_1 \wedge \varphi_2$, $\varphi_1 \Rightarrow \varphi_2$, etc.
  - We can use negation: $\neg\varphi_1$.

Quantifiers

- Quantifiers introduce new variables.

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier
- Given φ with free variable v, we can construct:

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier
- Given φ with free variable v, we can construct:
  - $\forall_{v \in V} \varphi$. This formula is true iff φ is true for *every* evaluation of v to a vertex of the graph.

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier
- Given φ with free variable v, we can construct:
  - $\forall_{v \in V} \varphi$. This formula is true iff φ is true for *every* evaluation of v to a vertex of the graph.
  - $\exists_{v \in V} \varphi$. This formula is true iff φ is true for *some* evaluation of v to a vertex of the graph.

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier
- Given φ with free variable v, we can construct:
  - $\forall_{v \in V} \varphi$. This formula is true iff φ is true for *every* evaluation of v to a vertex of the graph.
  - $\exists_{v \in V} \varphi$. This formula is true iff φ is true for *some* evaluation of v to a vertex of the graph.
- Think that the evaluation procedure branches into all possible evaluations and check whether every/some satisfies φ.

Quantifiers

- Quantifiers introduce new variables.
  - $\forall$: Universal quantifier
  - $\exists$: Existential quantifier
- Given φ with free variable v, we can construct:
  - $\forall_{v \in V} \varphi$. This formula is true iff φ is true for *every* evaluation of v to a vertex of the graph.
  - $\exists_{v \in V} \varphi$. This formula is true iff φ is true for *some* evaluation of v to a vertex of the graph.
- Think that the evaluation procedure branches into all possible evaluations and check whether every/some satisfies φ.
- Similarly we can quantify edges and vertex/edge subsets.

Syntactic sugar

- We shall use some shortcuts.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.
  - $X \subseteq Y$ is $\forall_{v \in V}\ (v \in X) \Rightarrow (v \in Y)$.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.
  - $X \subseteq Y$ is $\forall_{v \in V}\ (v \in X) \Rightarrow (v \in Y)$.
  - $\forall_{v \in X}\ \varphi$ is $\forall_{v \in V}\ (v \in X) \Rightarrow \varphi$.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.
  - $X \subseteq Y$ is $\forall_{v \in V}\ (v \in X) \Rightarrow (v \in Y)$.
  - $\forall_{v \in X}\ \varphi$ is $\forall_{v \in V}\ (v \in X) \Rightarrow \varphi$.
  - $\exists_{v \in X}\ \varphi$ is $\exists_{v \in V}\ (v \in X) \wedge \varphi$.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.
  - $X \subseteq Y$ is $\forall_{v \in V}\ (v \in X) \Rightarrow (v \in Y)$.
  - $\forall_{v \in X}\ \varphi$ is $\forall_{v \in V}\ (v \in X) \Rightarrow \varphi$.
  - $\exists_{v \in X}\ \varphi$ is $\exists_{v \in V}\ (v \in X) \wedge \varphi$.
  - $\mathbf{adj}(u, v)$ is $(u \neq v) \wedge \exists_{e \in E}\ (\mathbf{inc}(u, e) \wedge \mathbf{inc}(v, e))$.

Syntactic sugar

- We shall use some shortcuts.
  - $u \neq v$ is $\neg(u = v)$.
  - $X \subseteq Y$ is $\forall_{v \in V}\ (v \in X) \Rightarrow (v \in Y)$.
  - $\forall_{v \in X}\ \varphi$ is $\forall_{v \in V}\ (v \in X) \Rightarrow \varphi$.
  - $\exists_{v \in X}\ \varphi$ is $\exists_{v \in V}\ (v \in X) \wedge \varphi$.
  - $\mathbf{adj}(u, v)$ is $(u \neq v) \wedge \exists_{e \in E}\ (\mathbf{inc}(u, e) \wedge \mathbf{inc}(v, e))$.
- Other shortcuts possible.

Example 1

Sentence **3col** that is true in 3-colorable graphs.

Example 1

Sentence **3col** that is true in 3-colorable graphs.

$$\textbf{3col} = \quad \exists_{X_1,X_2,X_3 \subseteq V}\, \textbf{prt}(X_1, X_2, X_3) \wedge \textbf{indp}(X_1) \wedge \textbf{indp}(X_2) \wedge \textbf{indp}(X_3).$$

Example 1

Sentence **3col** that is true in 3-colorable graphs.

$$\textbf{3col} = \quad \exists_{X_1,X_2,X_3 \subseteq V}\, \textbf{prt}(X_1,X_2,X_3) \wedge \textbf{indp}(X_1) \wedge \textbf{indp}(X_2) \wedge \textbf{indp}(X_3).$$

$$\begin{aligned}\textbf{prt}(X_1,X_2,X_3) = \quad \forall_{v \in V}\, [&(v \in X_1 \wedge v \notin X_2 \wedge v \notin X_3) \\ &\vee (v \notin X_1 \wedge v \in X_2 \wedge v \notin X_3) \\ &\vee (v \notin X_1 \wedge v \notin X_2 \wedge v \in X_3)];\end{aligned}$$

Example 1

Sentence **3col** that is true in 3-colorable graphs.

$$\textbf{3col} = \quad \exists_{X_1,X_2,X_3 \subseteq V}\, \textbf{prt}(X_1, X_2, X_3) \wedge \textbf{indp}(X_1) \wedge \textbf{indp}(X_2) \wedge \textbf{indp}(X_3).$$

$$\textbf{prt}(X_1, X_2, X_3) = \quad \forall_{v \in V} [(v \in X_1 \wedge v \notin X_2 \wedge v \notin X_3) \vee (v \notin X_1 \wedge v \in X_2 \wedge v \notin X_3) \vee (v \notin X_1 \wedge v \notin X_2 \wedge v \in X_3)];$$

$$\textbf{indp}(X) = \quad \forall_{u,v \in X} \neg \textbf{adj}(u, v).$$

Example 2

Sentence **ham** that is true in graphs with a Hamiltonian cycle.

Example 2

Sentence **ham** that is true in graphs with a Hamiltonian cycle.

$$\textbf{ham} = \quad \exists_{C \subseteq E}\, \textbf{spans}(C) \wedge \forall_{v \in V}\, \textbf{deg2}(v, C).$$

Example 2

Sentence **ham** that is true in graphs with a Hamiltonian cycle.

$$\textbf{ham} = \quad \exists_{C \subseteq E}\, \textbf{spans}(C) \wedge \forall_{v \in V}\, \textbf{deg2}(v, C).$$

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

Example 2

Sentence **ham** that is true in graphs with a Hamiltonian cycle.

$$\textbf{ham} = \quad \exists_{C \subseteq E}\, \textbf{spans}(C) \wedge \forall_{v \in V}\, \textbf{deg2}(v, C).$$

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

$$\textbf{deg2}(v, C) = \quad \exists_{e_1, e_2 \in C}\, [(e_1 \neq e_2) \wedge \textbf{inc}(v, e_1) \wedge \textbf{inc}(v, e_2) \wedge (\forall_{e_3 \in C}\, \textbf{inc}(v, e_3) \Rightarrow (e_1 = e_3 \vee e_2 = e_3))]$$

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ, and

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ, and
- an n-vertex graph G with its tree decomposition of width t and evaluation of all the free variables of φ,

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ, and
- an n-vertex graph G with its tree decomposition of width t and evaluation of all the free variables of φ,

verifies whether φ is satisfied in G in time $f(||\varphi||, t) \cdot n$.

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ, and
- an n-vertex graph G with its tree decomposition of width t and evaluation of all the free variables of φ,

verifies whether φ is satisfied in G in time $f(||\varphi||, t) \cdot n$.

- **Corollary**: 3-colorability and Hamiltonicity can be checked in $f(t) \cdot n$ time.

Courcelle's theorem

Courcelle's theorem

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ, and
- an n-vertex graph G with its tree decomposition of width t and evaluation of all the free variables of φ,

verifies whether φ is satisfied in G in time $f(||\varphi||, t) \cdot n$.

- **Corollary**: 3-colorability and Hamiltonicity can be checked in $f(t) \cdot n$ time.
- **Note**: optimum treewidth can be computed in linear FPT time.

Some remarks

- The proof is ‘standard’.

Some remarks

- The proof is ‘standard’.
- A tree decomposition is turned into a rooted tree over an alphabet of size $f(t)$.

Some remarks

- The proof is 'standard'.
- A tree decomposition is turned into a rooted tree over an alphabet of size $f(t)$.
- The $\mathbf{MSO}_2$ formula is translated into a tree automaton of size $f(||\varphi||, t)$.

Some remarks

- The proof is 'standard'.
- A tree decomposition is turned into a rooted tree over an alphabet of size $f(t)$.
- The $\mathbf{MSO}_2$ formula is translated into a tree automaton of size $f(||\varphi||, t)$.
- The automaton is run on the tree.

Some remarks

- The proof is 'standard'.
- A tree decomposition is turned into a rooted tree over an alphabet of size $f(t)$.
- The $\mathbf{MSO}_2$ formula is translated into a tree automaton of size $f(||\varphi||, t)$.
- The automaton is run on the tree.
- Function f depends non-elementarily on $||\varphi||$ and t, and this is necessary.

Some remarks

- The proof is 'standard'.
- A tree decomposition is turned into a rooted tree over an alphabet of size $f(t)$.
- The $\mathbf{MSO}_2$ formula is translated into a tree automaton of size $f(||\varphi||, t)$.
- The automaton is run on the tree.
- Function f depends non-elementarily on $||\varphi||$ and t, and this is necessary.
- Roughly, the height of the tower in f is linear wrt. the quantifier alternation of φ.

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.
- Let's try to write it in $\mathbf{MSO}_2$.

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.
- Let's try to write it in $\mathbf{MSO}_2$.
- **Property**: the graph has a vertex cover of size $\leq k$.

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.
- Let's try to write it in $\mathbf{MSO}_2$.
- **Property**: the graph has a vertex cover of size $\leq k$.
- $\exists_{u_1 \in V} \ldots \exists_{u_k \in V} \forall_{e \in E} \mathbf{inc}(u_1, e) \vee \ldots \vee \mathbf{inc}(u_k, e)$.

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.
- Let's try to write it in $\mathbf{MSO}_2$.
- **Property**: the graph has a vertex cover of size $\leq k$.
- $\exists_{u_1 \in V} \ldots \exists_{u_k \in V} \forall_{e \in E} \mathbf{inc}(u_1, e) \vee \ldots \vee \mathbf{inc}(u_k, e)$.
- But this formula has length depending on k!

Next try: VC

- First lecture: $2^t \cdot t^{\mathcal{O}(1)} \cdot n$ algorithm for minimum vertex cover.
- Let's try to write it in $\mathbf{MSO}_2$.
- **Property**: the graph has a vertex cover of size $\leq k$.
- $\exists_{u_1 \in V} \ldots \exists_{u_k \in V} \forall_{e \in E} \mathbf{inc}(u_1, e) \vee \ldots \vee \mathbf{inc}(u_k, e)$.
- But this formula has length depending on k!
- Gives only an $f(k,t) \cdot n$ algorithm.

Optimization version

- We need sort of an optimization version of Courcelle's theorem.

Optimization version

- We need sort of an optimization version of Courcelle's theorem.
- Suppose $\varphi(X_1, X_2, \ldots, X_r)$ has r free monadic variables.

Optimization version

- We need sort of an optimization version of Courcelle's theorem.
- Suppose $\varphi(X_1, X_2, \ldots, X_r)$ has r free monadic variables.
- Let $\alpha(x_1, x_2, \ldots, x_r) = a_0 + \sum_{i=1}^{r} a_i x_i$ be an affine function.

Optimization version

Courcelle's theorem, optimization variant

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ with free monadic variables $X_1, X_2, \ldots, X_r$,
- an affine function $\alpha(x_1, x_2, \ldots, x_r)$, and
- an n-vertex graph G with its tree decomposition of width t,

finds the minimum/maximum of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$ over evaluations of $X_1, X_2, \ldots, X_r$ for which φ is satisfied. The running time is $f(||\varphi||, t) \cdot n$.

Optimization version

Courcelle's theorem, optimization variant

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ with free monadic variables $X_1, X_2, \ldots, X_r$,
- an affine function $\alpha(x_1, x_2, \ldots, x_r)$, and
- an n-vertex graph G with its tree decomposition of width t,

finds the minimum/maximum of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$ over evaluations of $X_1, X_2, \ldots, X_r$ for which φ is satisfied. The running time is $f(||\varphi||, t) \cdot n$.

- Vertex Cover: maximize $|X|$ subject to $\forall_{e \in E} \exists_{v \in X} \mathbf{inc}(v, e)$.

Optimization version

Courcelle's theorem, optimization variant

There exists an algorithm that, given

- an $\mathbf{MSO}_2$ formula φ with free monadic variables $X_1, X_2, \ldots, X_r$,
- an affine function $\alpha(x_1, x_2, \ldots, x_r)$, and
- an n-vertex graph G with its tree decomposition of width t,

finds the minimum/maximum of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$ over evaluations of $X_1, X_2, \ldots, X_r$ for which φ is satisfied. The running time is $f(||\varphi||, t) \cdot n$.

- Vertex Cover: maximize $|X|$ subject to $\forall_{e \in E} \exists_{v \in X} \mathbf{inc}(v, e)$.
- Gives an $f(t) \cdot n$ algorithm.

Remarks

- Formula φ can have free variables other than $X_1, X_2, \ldots, X_r$, whose evaluation is given together with the graph.

Remarks

- Formula φ can have free variables other than $X_1, X_2, \ldots, X_r$, whose evaluation is given together with the graph.
- Other variants and generalizations of Courcelle's theorem:

Remarks

- Formula φ can have free variables other than $X_1, X_2, \ldots, X_r$, whose evaluation is given together with the graph.
- Other variants and generalizations of Courcelle's theorem:
  - Exact value of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$, but a_i must be integers and the running time is quadratic in n.

Remarks

- Formula φ can have free variables other than $X_1, X_2, \ldots, X_r$, whose evaluation is given together with the graph.
- Other variants and generalizations of Courcelle's theorem:
  - Exact value of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$, but a_i must be integers and the running time is quadratic in n.
  - Counting the number of tuples $(X_1, X_2, \ldots, X_r)$ satisfying φ.

Remarks

- Formula φ can have free variables other than $X_1, X_2, \ldots, X_r$, whose evaluation is given together with the graph.
- Other variants and generalizations of Courcelle's theorem:
  - Exact value of $\alpha(|X_1|, |X_2|, \ldots, |X_r|)$, but a_i must be integers and the running time is quadratic in n.
  - Counting the number of tuples $(X_1, X_2, \ldots, X_r)$ satisfying φ.
  - Can use atomic formulas of the form $|X| \equiv a \mod p$, for constants a, p.

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.
- But it has big limitations:

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.
- But it has big limitations:
  - No good estimates on the running time.

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.
- But it has big limitations:
  - No good estimates on the running time.
  - Cannot handle real weights and other non-$\mathbf{MSO}_2$-expressible features.

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.
- But it has big limitations:
  - No good estimates on the running time.
  - Cannot handle real weights and other non-$\mathbf{MSO}_2$-expressible features.
- Good as a classification tool.

Conclusions

- Courcelle's theorem is a clean way to infer existence of DPs on a tree decomposition.
- But it has big limitations:
  - No good estimates on the running time.
  - Cannot handle real weights and other non-$\mathbf{MSO}_2$-expressible features.
- Good as a classification tool.
- If you want a precise running time bound, you need to get your hands dirty.