Étiquetage de Formules du Premier Ordre dans des graphes de Clique-width non Bornée.

M. M. KANTÉ B. COURCELLE C. GAVOILLE

Université Bordeaux 1, LaBRI, CNRS.

Groupe de Travail VAG (LIRMM)
13 Mars 2008

Introduction

- Let $P(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1,\ldots,a_m,W_1,\ldots,W_q)$ using $L(a_1),\ldots,L(a_m)$ et $L(W_1),\ldots,L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A},\mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1,\ldots,a_m,W_1,\ldots,W_q)$ using $L(a_1),\ldots,L(a_m)$ et $L(W_1),\ldots,L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A},\mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1, \ldots, a_m, W_1, \ldots, W_q)$ using $L(a_1), \ldots, L(a_m)$ et $L(W_1), \ldots, L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A}, \mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1,\ldots,a_m,W_1,\ldots,W_q)$ using $L(a_1),\ldots,L(a_m)$ et $L(W_1),\ldots,L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A},\mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1,\ldots,a_m,W_1,\ldots,W_q)$ using $L(a_1),\ldots,L(a_m)$ et $L(W_1),\ldots,L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A},\mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1,\ldots,a_m,W_1,\ldots,W_q)$ using $L(a_1),\ldots,L(a_m)$ et $L(W_1),\ldots,L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A},\mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1, \dots, x_m, Y_1, \dots, Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1, \dots, a_m, W_1, \dots, W_q)$ using $L(a_1), \dots, L(a_m)$ et $L(W_1), \dots, L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A}, \mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

- Let $P(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ be a graph property (adjacency, distance at most k, connectivity, ...)
- For a class of graphs $\mathcal{C}$, we want two algorithms $\mathcal{A}$ and $\mathcal{B}$ such that
  - For all $G \in \mathcal{C}$, $\mathcal{A}$, called *labeling algorithm*, constructs a labeling of the vertices of G,
  - $\mathcal{B}$, called *decoding algorithm*, checks whether G satisfies $P(a_1, \ldots, a_m, W_1, \ldots, W_q)$ using $L(a_1), \ldots, L(a_m)$ et $L(W_1), \ldots, L(W_q)$.
  - We require $\mathcal{B}$ independent from G, i.e., has to be the same for all $G \in \mathcal{C}$.
- The couple $(\mathcal{A}, \mathcal{B})$ is called *labeling scheme* (HERE).
- We want to minimize the length of the labels. We also require that the time complexity of $\mathcal{B}$ depends only on the length of the labels.
- We are interested in labeling schemes where the length of the labels are at most $O(\log^k(n))$ (k is fixed and n is (always) the number of vertices of graphs).

Introduction

Labeling schemes

Two approaches =>

1. P is fixed and we look for classes that accept labeling scheme with labels of length at most $O(f(n)) \ll O(n)$ (adjacency, distance for instance).
2. $\mathcal{C}$ is fixed and we look for problems expressible in logical languages like first-order (FO) or monadic second-order (MSO) logic such that there exist labeling schemes with labels of size $O(f(n)) \ll O(n)$.

Introduction

Lebeling schemes

Two approaches =>

1. P is fixed and we look for classes that accept labeling scheme with labels of length at most $O(f(n)) \ll O(n)$ (adjacency, distance for instance).
2. $\mathcal{C}$ is fixed and we look for problems expressible in logical languages like first-order (FO) or monadic second-order (MSO) logic such that there exist labeling schemes with labels of size $O(f(n)) \ll O(n)$.

- We are interested in this talk with *2.* and particularly with graphs with unbounded clique-width, particularly, the *locally cwd-decomposable classes*.
- Courcelle and Vanicat have already considered MSO queries on graphs of bounded clique-width.
- We are obliged to consider FO queries since the planar graphs are locally cwd-decomposable.

Plan

Plan

1. Clique-Width
2. Logic
3. Locally Decomposable Graphs
4. Main Results

Clique-Width

- A k-graph is a graph where the vertices are colored with colors in $\{1,\dots,k\}$. Each vertex with one color. We represent it by $\langle V_G, E_G, lab_G \rangle$.
- $G \oplus H$ is the disjoint union of G and H (notice that $G \oplus G \neq G$).
- $add_{ij}(G)$, $i \neq j$, is the graph $\langle V_G, E', lab_G \rangle$ where

$$E' = E_G \cup \{xy \mid lab_G(x) = i,\ lab_G(y) = j\}.$$

  This operation adds edges between vertices colored by i and vertices colored by j (a kind of complete bipartite graphs).
- $ren_{i \to j}(G)$ is the graph $\langle V_G, E_G, lab' \rangle$ where

$$lab'(x) = \begin{cases} j & \text{if } lab_G(x) = i \\ lab_G(x) & \text{otherwise} \end{cases}.$$

- **i** is the graph with single vertex colored by $i \in [k]$.

Clique-Width

- A k-graph is a graph where the vertices are colored with colors in $\{1,\ldots,k\}$. Each vertex with one color. We represent it by $\langle V_G, E_G, lab_G \rangle$.
- $G \oplus H$ is the disjoint union of G and H (notice that $G \oplus G \neq G$).
- $add_{ij}(G)$, $i \neq j$, is the graph $\langle V_G, E', lab_G \rangle$ where

$$E' = E_G \cup \{xy \mid lab_G(x) = i,\ lab_G(y) = j\}.$$

  This operation adds edges between vertices colored by i and vertices colored by j (a kind of complete bipartite graphs).
- $ren_{i \to j}(G)$ is the graph $\langle V_G, E_G, lab' \rangle$ where

$$lab'(x) = \begin{cases} j & \text{if } lab_G(x) = i \\ lab_G(x) & \text{otherwise} \end{cases}.$$

- $\mathbf{i}$ is the graph with single vertex colored by $i \in [k]$.

Clique-Width

- A k-graph is a graph where the vertices are colored with colors in $\{1,\ldots,k\}$. Each vertex with one color. We represent it by $\langle V_G, E_G, lab_G\rangle$.
- $G \oplus H$ is the disjoint union of G and H (notice that $G \oplus G \neq G$).
- $add_{ij}(G)$, $i \neq j$, is the graph $\langle V_G, E', lab_G\rangle$ where

$$E' = E_G \cup \{xy \mid lab_G(x) = i,\ lab_G(y) = j\}.$$

This operation adds edges between vertices colored by i and vertices colored by j (a kind of complete bipartite graphs).

- $ren_{i\to j}(G)$ is the graph $\langle V_G, E_G, lab'\rangle$ where

$$lab'(x) = \begin{cases} j & \text{if } lab_G(x) = i \\ lab_G(x) & \text{otherwise} \end{cases}.$$

- **i** is the graph with single vertex colored by $i \in [k]$.

Clique-Width

- A k-graph is a graph where the vertices are colored with colors in $\{1,\ldots,k\}$. Each vertex with one color. We represent it by $\langle V_G, E_G, lab_G \rangle$.
- $G \oplus H$ is the disjoint union of G and H (notice that $G \oplus G \neq G$).
- $add_{ij}(G)$, $i \neq j$, is the graph $\langle V_G, E', lab_G \rangle$ where

$$E' = E_G \cup \{xy \mid lab_G(x) = i,\ lab_G(y) = j\}.$$

  This operation adds edges between vertices colored by i and vertices colored by j (a kind of complete bipartite graphs).
- $ren_{i \to j}(G)$ is the graph $\langle V_G, E_G, lab' \rangle$ where

$$lab'(x) = \begin{cases} j & \text{if } lab_G(x) = i \\ lab_G(x) & \text{otherwise} \end{cases}.$$

- $\mathbf{i}$ is the graph with single vertex colored by $i \in [k]$.

Clique-Width

- A k-graph is a graph where the vertices are colored with colors in $\{1, \ldots, k\}$. Each vertex with one color. We represent it by $\langle V_G, E_G, lab_G \rangle$.
- $G \oplus H$ is the disjoint union of G and H (notice that $G \oplus G \neq G$).
- $add_{ij}(G)$, $i \neq j$, is the graph $\langle V_G, E', lab_G \rangle$ where

$$E' = E_G \cup \{xy \mid lab_G(x) = i,\ lab_G(y) = j\}.$$

  This operation adds edges between vertices colored by i and vertices colored by j (a kind of complete bipartite graphs).
- $ren_{i \to j}(G)$ is the graph $\langle V_G, E_G, lab' \rangle$ where

$$lab'(x) = \begin{cases} j & \text{if } lab_G(x) = i \\ lab_G(x) & \text{otherwise} \end{cases}.$$

- **i** is the graph with single vertex colored by $i \in [k]$.

Clique-Width

Well-Formed Terms

- $F_k = \{\oplus, add_{ij}, ren_{i \to j} \mid i, j \in [k], i \neq j\}$ and $C_k = \{\mathbf{i} \mid i \in [k]\}$.
- A term t defines, up to isomorphism, a graph $val(t)$ (we forget the colors).
- the clique-width of a graph G, denoted by $cwd(G)$, is the minimum k such that $G = val(t)$, $t \in T(F_k, C_k)$.
- bounded tree-width implies bounded clique-width but the converse is false (cliques have unbounded tree-width but clique-width 2)
- Examples => blackboard.

Clique-Width

Well-Formed Terms

- $F_k = \{\oplus, add_{ij}, ren_{i\to j} \mid i,j \in [k], i \neq j\}$ and $C_k = \{\mathbf{i} \mid i \in [k]\}$.
- A term t defines, up to isomorphism, a graph $val(t)$ (we forget the colors).
- the clique-width of a graph G, denoted by $cwd(G)$, is the minimum k such that $G = val(t)$, $t \in T(F_k, C_k)$.
- bounded tree-width implies bounded clique-width but the converse is false (cliques have unbounded tree-width but clique-width 2)
- Examples => blackboard.

Clique-Width

Well-Formed Terms

- $F_k = \{\oplus, add_{ij}, ren_{i\to j} \mid i,j \in [k], i \neq j\}$ and $C_k = \{\mathbf{i} \mid i \in [k]\}$.
- A term t defines, up to isomorphism, a graph $val(t)$ (we forget the colors).
- the clique-width of a graph G, denoted by $cwd(G)$, is the minimum k such that $G = val(t)$, $t \in T(F_k, C_k)$.
- bounded tree-width implies bounded clique-width but the converse is false (cliques have unbounded tree-width but clique-width 2)
- Examples => blackboard.

Clique-Width

Well-Formed Terms

- $F_k = \{\oplus, add_{ij}, ren_{i \to j} \mid i,j \in [k], i \neq j\}$ and $C_k = \{\mathbf{i} \mid i \in [k]\}$.
- A term t defines, up to isomorphism, a graph $val(t)$ (we forget the colors).
- the clique-width of a graph G, denoted by $cwd(G)$, is the minimum k such that $G = val(t)$, $t \in T(F_k, C_k)$.
- bounded tree-width implies bounded clique-width but the converse is false (cliques have unbounded tree-width but clique-width 2)
- Examples => blackboard.

Clique-Width

Some Results

- Every MSO query can be tested in graphs of clique-width at most k, k fixed.
- It uses tree-automata.
- The labeling scheme of Courcelle and Vanicat uses tree-automata and the fact that binary terms can be balanced.
- Follow explanations on blackboard.

Plan

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- Two sorts of variables: variables denoting vertices (lower case) and variables denoting subsets of vertices (capital letters).
- G is the structure $\langle V_G, E_G, P_{1G}, \ldots, P_{kG} \rangle$ where P_{iG} is an unary relation.
- $x = y$, $x \in X$, $E(x,y)$ and $P(x)$ are FO formulas.
- $\neg\varphi$, $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are FO formulas.
- $\exists x.\varphi(x)$ is a FO formula (x is in the scope of a quantifier).
- A free variable in a formula is a variable which is not inside the scope of a quantifier.
- We denote by $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ the FO formula φ with free FO variables in $\{x_1, \ldots, x_m\}$ and free MSO variables in $\{Y_1, \ldots, Y_q\}$.

FO Logic

- We write $G \models \varphi(a_1, \ldots, a_m, W_1, \ldots, W_q)$ to say that G satisfies $\varphi(a_1, \ldots, a_m, W_1, \ldots, W_q)$.
- An *FO sentence* is a FO formula without free variables.

Distance at most t

$$\varphi(x,y) := (x = y) \vee \bigvee_{1 \leq s \leq t} \left(\exists x_1. \cdots .\exists x_{s+1} \left(\bigwedge_{1 \leq i \leq t} E(x_i, x_{i+1}) \wedge x = x_1 \wedge y = x_s \right) \right).$$

Plan

Local Clique-Width

Classes of bounded Local Clique-Width

- The *local clique-width* of a graph G is the function $lcw^G : \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$lcw^G(t) := \max\{cwd(G[N_G^t(a)]) \mid a \in V_G\}.$$

- A class $\mathcal{C}$ of graphs has *bounded local clique-width* if there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $lcw^G(t) \leq f(t)$ for every $G \in \mathcal{C}$ and $t \in \mathbb{N}$.

Examples

Planar Graphs, unit-interval graphs, graphs of bounded degree, classes of bounded local tree-width . . . .

Locally cwd-decomposable

cwd-cover

Let $r, l \geq 1$ and $g : \mathbb{N} \rightarrow \mathbb{N}$. An *$(r,l,g)$-cwd cover* of a graph G is a family $\mathcal{T}$ of subsets of V_G such that:

1. For every $a \in V_G$ there exists a $U \in \mathcal{T}$ such that $N_G^r(a) \subseteq U$.
2. For each $U \in \mathcal{T}$ there exist less than l many $V \in \mathcal{T}$ such that $U \cap V \neq \emptyset$.
3. For each U we have $cwd(G[U]) \leq g(1)$.

Locally cwd-decomposable

cwd-cover

Let $r,l \geq 1$ and $g : \mathbb{N} \to \mathbb{N}$. An *$(r,l,g)$-cwd cover* of a graph G is a family $\mathcal{T}$ of subsets of V_G such that:

1. For every $a \in V_G$ there exists a $U \in \mathcal{T}$ such that $N^r_G(a) \subseteq U$.
2. For each $U \in \mathcal{T}$ there exist less than l many $V \in \mathcal{T}$ such that $U \cap V \neq \emptyset$.
3. For each U we have $cwd(G[U]) \leq g(1)$.

Nice cwd-cover

An (r,l,g)-cwd cover is *nice* if condition 3 is replaced by condition 3' below:

3'. For all $U_1,\ldots,U_q$ and $q \geq 1$ we have

$$cwd(G[U_1 \cup \cdots \cup U_q]) \leq g(q).$$

Locally cwd-decomposable

Locally cwd-decomposable

A class $\mathcal{C}$ of graphs is *locally cwd-decomposable* if there is a polynomial time algorithm that given a graph $G \in \mathcal{C}$ and $r \geq 1$, computes an (r, l, g)-cwd cover of G for suitable l, g depending on r.

Locally cwd-decomposable

Locally cwd-decomposable

A class $\mathcal{C}$ of graphs is *locally cwd-decomposable* if there is a polynomial time algorithm that given a graph $G \in \mathcal{C}$ and $r \geq 1$, computes an (r, l, g)-cwd cover of G for suitable l, g depending on r.

Nicely locally cwd-decomposable

A class $\mathcal{C}$ of graphs is *nicely locally cwd-decomposable* if there is a polynomial time algorithm that given a graph $G \in \mathcal{C}$ and $r \geq 1$, computes a nice (r, l, g)-cwd cover of G for suitable l, g depending on r.

Locally cwd-decomposable

Locally cwd-decomposable

A class $\mathcal{C}$ of graphs is *locally cwd-decomposable* if there is a polynomial time algorithm that given a graph $G \in \mathcal{C}$ and $r \geq 1$, computes an (r, l, g)-cwd cover of G for suitable l, g depending on r.

Nicely locally cwd-decomposable

A class $\mathcal{C}$ of graphs is *nicely locally cwd-decomposable* if there is a polynomial time algorithm that given a graph $G \in \mathcal{C}$ and $r \geq 1$, computes a nice (r, l, g)-cwd cover of G for suitable l, g depending on r.

Fact

- Nicely locally cwd-decomposable implies locally cwd-decomposable.
- locally cwd-decomposable implies local bounded clique-width.

Plan

Main Results

Main Theorem 1

There exist $O(\log(n))$-labeling schemes for the following queries and graph classes:

1. FO queries without set arguments on locally cwd-decomposable classes.
2. FO queries with set arguments on nicely locally cwd-decomposable.

FO Logic

t-local formulas

An FO formula $\varphi(x_1,\ldots,x_m,Y_1,\ldots,Y_q)$ is *t-local around* $(x_1,\ldots,x_m)$ if for every G and, every $a_1,\ldots,a_m \in V_G,\ W_1,\ldots,W_q \subseteq V_G$ we have

$$G \models \varphi(a_1,\ldots,a_m,W_1,\ldots,W_q)$$

iff

$$G[N] \models \varphi(a_1,\ldots,a_m,W_1\cap N,\ldots,W_q\cap N)$$

where $N = N_G^t(a_1,\ldots,a_m) = \{y \in V_G \mid d(y,a_i) \leq t \text{ for some } i = 1,\ldots,m\}$.

FO Logic

t-local formulas

An FO formula $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ is *t-local around* $(x_1, \ldots, x_m)$ if for every G and, every $a_1, \ldots, a_m \in V_G,\ W_1, \ldots, W_q \subseteq V_G$ we have

$$G \models \varphi(a_1, \ldots, a_m, W_1, \ldots, W_q)$$

iff

$$G[N] \models \varphi(a_1, \ldots, a_m, W_1 \cap N, \ldots, W_q \cap N)$$

where $N = N_G^t(a_1, \ldots, a_m) = \{y \in V_G \mid d(y, a_i) \leq t \text{ for some } i = 1, \ldots, m\}$.

Remark

The query $d(x,y) \leq r$ is t-local with $t = r/2$ if r is even and $(r-1)/2$ if r is odd. Its negation $d(x,y) > r$ is t-local

FO Logic

(t,s)-local sentences

An FO sentence is *basic (t,s)-local* if it is equivalent to a sentence of the form

$$\exists x_1.\cdots.\exists x_s.\left(\bigwedge_{1\leq i<j\leq s} d(x_i,x_j) > 2t \wedge \bigwedge_{1\leq i\leq s} \psi(x_i)\right)$$

where $\psi(x)$ is t-local around its unique free variable x.

Gaifman Theorem

Theorem 1

Let $\varphi(\overline{x})$ be a FO formula where $\overline{x} = (x_1, \dots, x_m)$. Then φ is logically equivalent to a Boolean combination $B(\varphi_1(\overline{u_1}), \dots, \varphi_p(\overline{u_p}), \psi_1, \dots, \psi_h)$ where:

- each $(\varphi_i)_{1 \leq i \leq p}$ is a t-local formula around $\overline{u_i} \subseteq \overline{x}$.
- each $(\psi_i)_{1 \leq i \leq h}$ is a basic (t', s)-local sentence.

Moreover B can be computed effectively and, t, t' and s can be bounded in terms of m and the quantifier-rank of φ.

Verification of basic (t,s)-local sentences

Lemma 1

Let G be in a locally cwd-decomposable class. Every basic (t,s)-local sentence without free set arguments can be decided in polynomial time.

Proof. Let φ be a sentence :

$$\exists x_1.\cdots.\exists x_s.\left(\bigwedge_{1\leq i<j\leq s} d(x_i,x_j) > 2t \wedge \bigwedge_{1\leq i\leq s} \psi(x_i)\right)$$

Proof(1)

Proof.

1. Let $\mathcal{T}$ be an (t,l,g)-cwd cover of G.
2. For each $U \in \mathcal{T}$ let $P_U = \{a \mid N_G^t(a) \subseteq U, G[N_G^t(a)] \models \psi(a)\}$.
3. Let $P = \bigcup_{U \in \mathcal{T}} P_U$.
4. If there exists $a_1, \ldots, a_s$ in P such that $d(a_i, a_j) > 2t,\ 1 \leq i < j \leq s$ then return TRUE.
5. Otherwise return FALSE.

Proof(1)

Correctness

- Line 1 can be done in polynomial time (G locally cwd-decomposable)
- For each U we can compute in polynomial time the set $K^t(U) := \{a \mid N^t_G(a) \subseteq U\}$.
- By Courcelle and Oum, for each $a \in K^t(U)$ we can test if $U \models \psi(a)$.
- Then Line 2 can be computed in polynomial time.
- Line 4. can be done in polynomial time in the size $|G| = O(n^2)$, of G (next slide).

Proof(2)

Input: G, P.
Output: Decide if there exists $a_1, \dots, a_s$ in P such that $d(a_i, a_j) > t$.

Algorithm

- Choose the $p \leq s$ vertices such that $P \subseteq N_G^t(a_1, \dots, a_p)$.
- If $p = m$ return YES.
- If $p = 0$, return NO.
- Otherwise computes $H = G[N_G^{2t}(a_1, \dots, a_p)]$.
- Let

$$\theta := \exists x_1. \cdots .\exists x_s. \left(\bigwedge_{1 \leq i < j \leq s} d(x_i, x_j) > 2t \ \wedge \bigwedge_{1 \leq i \leq s} x_i \in P \right).$$

- If $G[H] \models \theta$ then return YES.
- Otherwise return NO.

t-local formulas (stronger statement)

Lemma 2

There exists an $O(\log(n))$-labeling scheme for t-local formulas with set arguments on locally cwd-decomposable classes.

Proof.

- We will use a decomposition of t-local formulas by Frick.
- We recall that Gaifman Theorem extends to FO formulas with set arguments.
- It is not natural but is powerful enough for our purposes.

t-distance type

Definition 1

Let $m,t \geq 1$. The *t-distance type* of an m-tuple $\overline{a}$ is the undirected graph $\varepsilon = ([m], edg_\varepsilon)$ where $edg_\varepsilon(i,j)$ iff $d(a_i,a_j) \leq 2t+1$.

Satisfaction

The satisfaction of a t-distance type by an m-tuple can be expressed by a t-local formula:

$$\rho_{t,\varepsilon}(x_1,\ldots,x_m) := \bigwedge_{(i,j)\in edg_\varepsilon} d(x_i,x_j) \leq 2t+1 \;\wedge \bigwedge_{(i,j)\notin edg_\varepsilon} d(x_i,x_j) > 2t+1.$$

Decomposition of t-local formulas

Lemma 3

Let $\varphi(\overline{x}, Y_1, \ldots, Y_q)$ be a t-local formula around $\overline{x} = (x_1, \ldots, x_m)$, $m \geq 1$. For each t-distance type ε with $\varepsilon_1, \ldots, \varepsilon_p$ as connected components, one can compute a Boolean combination $F^{t,\varepsilon}(\varphi_{1,1}, \ldots, \varphi_{1,j_1}, \ldots, \varphi_{p,1}, \ldots, \varphi_{p,j_p})$ of formulas $\varphi_{i,j}$ such that:

- The FO free variables of each $\varphi_{i,j}$ are among $\overline{x} \mid \varepsilon_i$ ($\overline{x} \mid \varepsilon_i$ is the restriction of $\overline{x}$ to ε_i) and the set arguments remains in $\{Y_1, \ldots, Y_q\}$.
- $\varphi_{i,j}$ is t-local around $\overline{x} \mid \varepsilon_i$.
- For each m-tuple $\overline{a}$, each q-tuple of sets $W_1, \ldots, W_q$:

$$G \models \rho_{t,\varepsilon}(\overline{a}) \wedge \varphi(\overline{a}, W_1, \ldots, W_q)$$

iff

$$G \models \rho_{t,\varepsilon}(\overline{a}) \wedge F^{t,\varepsilon}(\ldots, \varphi_{i,j}(\overline{a} \mid \varepsilon_i, W_1, \ldots, W_q), \ldots).$$

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\ldots,a_s,W_1,\ldots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \ldots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\dots,a_s,W_1,\dots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \dots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\dots,a_s,W_1,\dots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \dots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r, l, g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x, y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1, \ldots, a_s, W_1, \ldots, W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \ldots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\ldots,a_s,W_1,\ldots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \ldots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\dots,a_s,W_1,\dots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \dots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2

- Let $\mathcal{T}$ be an (r,l,g)-cwd cover of G where $r = m(2t+1)$.
- Each $x \in V_G$ is in less than l many $V \in \mathcal{T}$.
- By Courcelle and Vanicat we can label each vertex with a label $K(x)$ of length $O(\log(n))$ and decide if $d(x,y) \leq 2t+1$ in $O(\log(n))$-time by using $K(x)$ and $K(y)$.
- For each $U \in \mathcal{T}$ and each $\varphi_{i,j}$, we can label each vertex $x \in U$ with a label $J^{\varepsilon}_{i,j,U}(x)$ and decide $\varphi_{i,j}(a_1,\ldots,a_s,W_1,\ldots,W_q)$ by using only $J^{\varepsilon}_{i,j,U}(a_i)$ and $J^{\varepsilon}_{i,j,U}(W_i \cap U)$.
- We do the same for all $\varphi_{i,j}$.
- For each x we append all these labels $J^{\varepsilon}_{i,j,U}$ in order to get a label J_{ε}.
- There exists at most $k' = 2^{k(k-1)/2}$ t-distance types, we let

$$J(x) = \{\ulcorner x \urcorner, K(x), J_{\varepsilon^1}, \ldots, J_{\varepsilon^{k'}}\}.$$

- It has length $O(\log(n))$ (Huge Constants).

Proof of Lemma 2(end)

- Let $J(a_1),\ldots,J(a_m)$ and $J(W_1),\ldots,J(W_q)$.
- By using $K(a_i)$ we can construct the t-distance type ε satisfied by $a_1,\ldots,a_m$. We can then recover $J_\varepsilon(a_i)$.
- We let $\varepsilon_1,\ldots,\varepsilon_p$ be the connected components of ε.
- For each $\bar{a} \mid \varepsilon_i$ there exists at least one $U \in \mathcal{T}$ such that $N_G^t(\bar{a} \mid \varepsilon_i) \subseteq U$. (There are less than l.)
- We can now decide whether G satisfies φ by Lemma 3.

Proof of Lemma 2(end)

- Let $J(a_1),\ldots,J(a_m)$ and $J(W_1),\ldots,J(W_q)$.
- By using $K(a_i)$ we can construct the t-distance type ε satisfied by $a_1,\ldots,a_m$. We can then recover $J_\varepsilon(a_i)$.
- We let $\varepsilon_1,\ldots,\varepsilon_p$ be the connected components of ε.
- For each $\overline{a} \mid \varepsilon_i$ there exists at least one $U \in \mathcal{T}$ such that $N^t_G(\overline{a} \mid \varepsilon_i) \subseteq U$. (There are less than l.)
- We can now decide whether G satisfies φ by Lemma 3.

Proof of Lemma 2(end)

- Let $J(a_1),\ldots,J(a_m)$ and $J(W_1),\ldots,J(W_q)$.
- By using $K(a_i)$ we can construct the t-distance type ε satisfied by $a_1,\ldots,a_m$. We can then recover $J_\varepsilon(a_i)$.
- We let $\varepsilon_1,\ldots,\varepsilon_p$ be the connected components of ε.
- For each $\overline{a} \mid \varepsilon_i$ there exists at least one $U \in \mathcal{T}$ such that $N_G^t(\overline{a} \mid \varepsilon_i) \subseteq U$. (There are less than l.)
- We can now decide whether G satisfies φ by Lemma 3.

Proof of Lemma 2(end)

- Let $J(a_1),\ldots,J(a_m)$ and $J(W_1),\ldots,J(W_q)$.
- By using $K(a_i)$ we can construct the t-distance type ε satisfied by $a_1,\ldots,a_m$. We can then recover $J_\varepsilon(a_i)$.
- We let $\varepsilon_1,\ldots,\varepsilon_p$ be the connected components of ε.
- For each $\overline{a} \mid \varepsilon_i$ there exists at least one $U \in \mathcal{T}$ such that $N_G^t(\overline{a} \mid \varepsilon_i) \subseteq U$. (There are less than l.)
- We can now decide whether G satisfies φ by Lemma 3.

Proof of Lemma 2(end)

- Let $J(a_1),\dots,J(a_m)$ and $J(W_1),\dots,J(W_q)$.
- By using $K(a_i)$ we can construct the t-distance type ε satisfied by $a_1,\dots,a_m$. We can then recover $J_\varepsilon(a_i)$.
- We let $\varepsilon_1,\dots,\varepsilon_p$ be the connected components of ε.
- For each $\bar{a} \mid \varepsilon_i$ there exists at least one $U \in \mathcal{T}$ such that $N_G^t(\bar{a} \mid \varepsilon_i) \subseteq U$. (There are less than l.)
- We can now decide whether G satisfies φ by Lemma 3.

Perspectives

- The tightness of the labels.
- Can we extend the results to local bounded clique-width classes ?
- Can we extend the logic ?

Perspectives

- The tightness of the labels.
- Can we extend the results to local bounded clique-width classes ?
- Can we extend the logic ?

Perspectives

- The tightness of the labels.
- Can we extend the results to local bounded clique-width classes ?
- Can we extend the logic ?

Perspectives

- The tightness of the labels.
- Can we extend the results to local bounded clique-width classes ?
- Can we extend the logic ?
- Thank you !