Logical Methods in Computer Science
Volume 15, Issue 3, 2019, pp. 2:1–2:34
https://lmcs.episciences.org/

Submitted Mar. 01, 2018
Published Jul. 08, 2019

CAPTURING LOGARITHMIC SPACE AND POLYNOMIAL TIME ON CHORDAL CLAW-FREE GRAPHS

BERIT GRUSSIEN

Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
e-mail address: grussien@informatik.hu-berlin.de

Abstract. We show that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization. $\mathsf{LREC}_=$ is a logic that extends first-order logic with counting by an operator that allows it to formalize a limited form of recursion. This operator can be evaluated in logarithmic space. It follows that there exists a logarithmic-space canonization algorithm, and therefore a logarithmic-space isomorphism test, for the class of chordal claw-free graphs. As a further consequence, $\mathsf{LREC}_=$ captures logarithmic space on this graph class. Since $\mathsf{LREC}_=$ is contained in fixed-point logic with counting, we also obtain that fixed-point logic with counting captures polynomial time on the class of chordal claw-free graphs.

1. Introduction

Descriptive complexity is a field of computational complexity theory that provides logical characterizations for the standard complexity classes. The starting point of descriptive complexity was a theorem of Fagin in 1974 [Fag74], which states that existential second-order logic characterizes, or *captures*, the complexity class NP. Later, similar logical characterizations were found for further complexity classes. For example, Immerman proved that deterministic transitive closure logic DTC captures LOGSPACE [Imm87], and independently of one another, Immerman [Imm86] and Vardi [Var82] showed that fixed-point logic FP captures PTIME[1]. However, these two results have a draw-back: They only hold on ordered structures, that is, on structures with a distinguished binary relation which is a linear order on the universe of the structure. On structures that are not necessarily ordered, there exist only partial results towards capturing LOGSPACE or PTIME.

A negative partial result towards capturing LOGSPACE follows from Etessami and Immerman's result that (directed) tree isomorphism is not definable in transitive closure logic with counting TC+C [EI00]. This implies that tree isomorphism is neither definable in deterministic nor symmetric transitive closure logic with counting (DTC+C and STC+C), although it is decidable in LOGSPACE [Lin92]. Hence, DTC+C and STC+C are not strong enough to capture LOGSPACE even on the class of trees. That is why, in 2011 a new logic

Key words and phrases: Chordal claw-free graphs, descriptive complexity, canonization, isomorphism problem, logarithmic space, polynomial time, fixed-point logic.

* This article is an extended version of [Gru17a].

1 More precisely, Immerman and Vardi's theorem holds for least fixed-point logic (LFP) and the equally expressive inflationary fixed-point logic (IFP). Our indeterminate FP refers to either of these two logics.

DOI:10.23638/LMCS-15(3:2)2019

with logarithmic-space data complexity was introduced [GGHL11, GGHL12]. This logic, $\mathsf{LREC}_=$, is an extension of first-order logic with counting by an operator that allows a limited form of recursion. $\mathsf{LREC}_=$ strictly contains $\mathsf{STC}{+}\mathsf{C}$ and $\mathsf{DTC}{+}\mathsf{C}$. In [GGHL11, GGHL12], the authors proved that $\mathsf{LREC}_=$ captures LOGSPACE on the class of (directed) trees and on the class of interval graphs. In this paper we now show that $\mathsf{LREC}_=$ captures LOGSPACE also on the class of chordal claw-free graphs, i.e., the class of all graphs that do not contain a cycle of length at least 4 (chordal) or the complete bipartite graph $K_{1,3}$ (claw-free) as an induced subgraph. More precisely, this paper's main technical contribution states that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization. This does not only imply that $\mathsf{LREC}_=$ captures LOGSPACE on chordal claw-free graphs, but also that there exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs. Hence, the isomorphism problem for this graph class is solvable in logarithmic space.

For polynomial time there also exist partial characterizations. Fixed-point logic with counting $\mathsf{FP}{+}\mathsf{C}$ captures PTIME, for example, on planar graphs [Gro98], on all classes of graphs of bounded treewidth [GM99] and on K_5-minor free graphs [Gro08]. Note that all these classes can be defined by a list of forbidden minors. In fact, Grohe showed in 2010 that $\mathsf{FP}{+}\mathsf{C}$ captures PTIME on all graph classes with excluded minors [Gro10b]. Instead of graph classes with excluded minors, one can also consider graph classes with excluded induced subgraphs, i.e., graph classes $\mathcal{C}$ that are closed under taking induced subgraphs. For some of these graph classes $\mathcal{C}$, e.g., chordal graphs [Gro10a], comparability graphs [Lau11] and co-comparability graphs [Lau11], capturing PTIME on $\mathcal{C}$ is as hard as capturing PTIME on the class of all graphs for any "reasonable" logic.[2] This gives us reason to consider subclasses of chordal graphs, comparability graphs and co-comparability graphs more closely. There are results showing that $\mathsf{FP}{+}\mathsf{C}$ captures PTIME on interval graphs (chordal co-comparability graphs) [Lau10], on permutation graphs (comparability co-comparability graphs) [Gru17c] and on chordal comparability graphs [Gru17b]. Further, Grohe proved that $\mathsf{FP}{+}\mathsf{C}$ captures PTIME on chordal line graphs [Gro10a]. At the same time he conjectured that this is also the case for the class of chordal claw-free graphs, which is an extension of the class of chordal line graphs. Our main result implies that Grohe's conjecture is true: Since $\mathsf{LREC}_=$ is contained in $\mathsf{FP}{+}\mathsf{C}$, it yields that there exists an $\mathsf{FP}{+}\mathsf{C}$-canonization of the class of chordal claw-free graphs. Hence, $\mathsf{FP}{+}\mathsf{C}$ captures PTIME also on the class of chordal claw-free graphs.

Our main result is based on a study of chordal claw-free graphs. Chordal graphs are the intersection graphs of subtrees of a tree [Bun74, Gav74, Wal72], and a clique tree of a chordal graph corresponds to a minimal representation of the graph as such an intersection graph. We prove that chordal claw-free graphs are (claw-free) intersection graphs of paths in a tree, and that for each connected chordal claw-free graph the clique tree is unique.

Structure. The preliminaries in Section 2 will be followed by a Section 3 where we analyze the structure of clique trees of chordal claw-free graphs, and, e.g., show that connected chordal claw-free graphs have a unique clique tree. In Section 4, we transform the clique tree of a connected chordal claw-free graph into a directed tree, and color each maximal clique with information about its intersection with other maximal cliques by using a special coloring with a linearly ordered set of colors. We obtain what we call the supplemented clique tree,

[2] Note that $\mathsf{FP}{+}\mathsf{C}$ does not capture PTIME on the class of all graphs [CFI92]. Hence, it does not capture PTIME on the class of chordal graphs, comparability graphs or co-comparability graphs either.

and show that it is definable in STC+C by means of a parameterized transduction. We know that there exists an $\mathsf{LREC}_=$-canonization of colored directed trees if the set of colors is linearly ordered [GGHL11, GGHL12]. In Section 5, we apply this $\mathsf{LREC}_=$-canonization to the supplemented clique tree and obtain the canon of this colored directed tree. Due to the type of coloring, the information about the maximal cliques is also contained in the colors of the canon of the supplemented clique tree. This information and the linear order on the vertices of the canon of the supplemented clique tree allow us to define the maximal cliques of a canon of the connected chordal claw-free graph, from which we can easily construct the canon of the graph. By combining the canons of the connected components, we obtain a canon for each chordal claw free graph. Finally, we present consequences of this canonization result in Section 6 and conclude in Section 7.

2. Basic Definitions and Notation

We write $\mathbb{N}$ for the set of all non-negative integers. For all $n, n' \in \mathbb{N}$, we define $[n, n'] := \{m \in \mathbb{N} \mid n \leq m \leq n'\}$ and $[n] := [1, n]$. We often denote tuples $(a_1, \ldots, a_k)$ by $\bar{a}$. Given a tuple $\bar{a} = (a_1, \ldots, a_k)$, let $\tilde{a} := \{a_1, \ldots, a_k\}$. Let $n \geq 1$. Let $\bar{a}^i = (a^i_1, \ldots, a^i_{k_i})$ be a tuple of length k_i for each $i \in [n]$. We denote the tuple $(a^1_1, \ldots, a^1_{k_1}, \ldots, a^n_1, \ldots, a^n_{k_n})$ by $(\bar{a}^1, \ldots, \bar{a}^n)$. Mappings $f \colon A \to B$ are extended to tuples $\bar{a} = (a_1, \ldots, a_k)$ over A via $f(\bar{a}) := (f(a_1), \ldots, f(a_k))$. Let $\approx$ be an equivalence relation on a set S. Then $a/_{\approx}$ denotes the equivalence class of $a \in S$ with respect to $\approx$. For $\bar{a} = (a_1, \ldots, a_n) \in S^n$ and $R \subseteq S^n$, we let $\bar{a}/_{\approx} := (a_1/_{\approx}, \ldots, a_n/_{\approx})$ and $R/_{\approx} := \{\bar{a}/_{\approx} \mid \bar{a} \in R\}$. A *partition* of a set S is a set $\mathcal{P}$ of disjoint non-empty subsets of S where $S = \bigcup_{A \in \mathcal{P}} A$. For a set S, we let $\binom{S}{2}$ be the set of all 2-element subsets of S.

2.1. Graphs and LO-Colorings.

A *graph* is a pair (V, E) consisting of a non-empty finite set V of *vertices* and a set $E \subseteq \binom{V}{2}$ of *edges*. Let $G = (V, E)$ and $G' = (V', E')$ be graphs. The *union* $G \cup G'$ of G and G' is the graph $(V \cup V', E \cup E')$. For a subset $W \subseteq V$ of vertices, $G[W]$ denotes the *induced subgraph* of G with vertex set W. *Connectivity* and *connected components* are defined in the usual way. We denote the *neighbors* of a vertex $v \in V$ by $N(v)$. A set $B \subseteq V$ is a *clique* if $\binom{B}{2} \subseteq E$. A maximal clique, or *max clique*, is a clique that is not properly contained in any other clique.

A graph is *chordal* if all its cycles of length at least 4 have a chord, which is an edge that connects two non-consecutive vertices of the cycle. A *claw-free* graph is a graph that does not have a *claw*, i.e., a graph isomorphic to the complete bipartite graph $K_{1,3}$, as an induced subgraph. We denote the class of (connected) chordal claw-free graphs by (con-)CCF.

A subgraph P of G is a *path* of G if $P = (\{v_0, \ldots, v_k\}, \{\{v_0, v_1\}, \ldots, \{v_{k-1}, v_k\}\})$ for distinct vertices $v_0, \ldots, v_k$ of G. We also denote the path P by the sequence $v_0, \ldots, v_k$ of vertices. We let v_0 and v_k be the *ends* of P. A connected acyclic graph is a *tree*. Let $T = (V, E)$ be a tree. A *subtree* of T is a connected subgraph of T. A vertex $v \in V$ of degree 1 is called a *leaf*.

A pair (V, E) is a *directed graph* or *digraph* if V is a non-empty finite set and $E \subseteq V^2$. A *path* of a digraph $G = (V, E)$ is a directed subgraph $P = (\{v_0, \ldots, v_k\}, \{(v_0, v_1), \ldots, (v_{k-1}, v_k)\})$ of G where the vertices $v_0, \ldots, v_k \in V$ are distinct. A connected acyclic digraph where the in-degree of each vertex is at most 1 is a *directed tree*. Let $T = (V, E)$ be a directed tree. The vertex of in-degree 0 is the *root* of T. If $(v, w) \in E$, then w is a *child* of v, and v the *parent* of w. Let w, w' be children of $v \in V$. Then w is a *sibling* of w' if $w \neq w'$. If there is a path from $v \in V$ to $w \in V$ in T, then v is an *ancestor* of w.

Let $G = (V, E)$ be a digraph and $f\colon V \to C$ be a mapping from the vertices of G to a finite set C. Then f is a *coloring* of G, and the elements of C are called *colors*. In this paper we color the vertices of a digraph with binary relations on a linearly ordered set. We call digraphs with such a coloring LO-*colored digraphs*. More precisely, an LO-colored digraph is a tuple $G = (V, E, M, \trianglelefteq, L)$ with the following four properties:

(1) The pair (V, E) is a digraph. We call (V, E) the *underlying digraph* of G.
(2) The set of *basic color elements* M is a non-empty finite set with $M \cap V = \emptyset$.
(3) The binary relation $\trianglelefteq \subseteq M^2$ is a linear order on M.
(4) The ternary relation $L \subseteq V \times M^2$ assigns to every vertex $v \in V$ an LO-*color* $L_v := \{(d, d') \mid (v, d, d') \in L\}$.

We can use the linear order $\trianglelefteq$ on M to obtain a linear order on the colors $\{L_v \mid v \in V\}$ of G. Thus, an LO-colored digraph is a special kind of colored digraph with a linear order on its colors.

2.2. **Structures.** A *vocabulary* is a finite set τ of relation symbols. Each relation symbol $R \in \tau$ has a fixed arity $\mathrm{ar}(R) \in \mathbb{N}$. A τ-*structure* A consists of a non-empty finite set $U(A)$, its *universe*, and for each relation symbol $R \in \tau$ of a relation $R(A) \subseteq U(A)^{\mathrm{ar}(R)}$.

An *isomorphism* between τ-structures A and B is a bijection $f\colon U(A) \to U(B)$ such that for all $R \in \tau$ and all $\bar{a} \in U(A)^{\mathrm{ar}(R)}$ we have $\bar{a} \in R(A)$ if and only if $f(\bar{a}) \in R(B)$. We write $A \cong B$ to indicate that A and B are *isomorphic*.

Let E be a binary relation symbol. Each graph corresponds to an $\{E\}$-structure $G = (V, E)$ where the universe V is the vertex set and E is an irreflexive and symmetric binary relation, the edge relation. Similarly, a digraph is represented by an $\{E\}$-structure $G = (V, E)$ where V is the vertex set and the edge relation E is an irreflexive binary relation. To represent an LO-colored digraph $G = (V, E, M, \trianglelefteq, L)$ as a logical structure, we extend the 5-tuple by a set U to a 6-tuple $(U, V, E, M, \trianglelefteq, L)$, and we require that $U = V \mathbin{\dot\cup} M$ in addition to the properties 1–4. The set U serves as the universe of the structure, and $V, E, M, \trianglelefteq, L$ are relations on U. We usually do not distinguish between (LO-colored) digraphs and their representation as logical structures. It will be clear from the context which form we are referring to.

2.3. **Logics.** In this section we introduce first-order logic with counting, symmetric transitive closure logic (with counting) and the logic $\mathsf{LREC}_=$. We assume basic knowledge in logic, in particular of *first-order logic* (FO).

First-order logic with counting ($\mathsf{FO{+}C}$*)* extends FO by a counting operator that allows for counting the cardinality of $\mathsf{FO{+}C}$-definable relations. It lives in a two-sorted context, where structures A are equipped with a *number sort* $N(A) := [0, |U(A)|]$. $\mathsf{FO{+}C}$ has two types of variables: $\mathsf{FO{+}C}$-variables are either *structure variables* that range over the universe $U(A)$ of a structure A, or *number variables* that range over the number sort $N(A)$. For each variable u, let $A^u := U(A)$ if u is a structure variable, and $A^u := N(A)$ if u is a number variable. Let $A^{(u_1,\dots,u_k)} := A^{u_1} \times \cdots \times A^{u_k}$. Tuples $(u_1, \dots, u_k)$ and $(v_1, \dots, v_\ell)$ of variables are *compatible* if $k = \ell$, and for every $i \in [k]$ the variables u_i and v_i are of the same type. An *assignment in A* is a mapping α from the set of variables to $U(A) \cup N(A)$, where for each variable u we have $\alpha(u) \in A^u$. For tuples $\bar{u} = (u_1, \dots, u_k)$ of variables and $\bar{a} = (a_1, \dots, a_k) \in A^{\bar{u}}$, the assignment $\alpha[\bar{a}/\bar{u}]$ maps u_i to a_i for each $i \in [k]$, and each variable $v \notin \tilde{u}$ to $\alpha(v)$. By $\varphi(u_1, \dots, u_k)$ we denote a formula φ with $\mathrm{free}(\varphi) \subseteq \{u_1, \dots, u_k\}$, where $\mathrm{free}(\varphi)$ is the set of free variables in φ. Given a formula $\varphi(u_1, \dots, u_k)$, a structure A and $(a_1, \dots, a_k) \in A^{(u_1,\dots,u_k)}$, we write $A \models \varphi[a_1, \dots, a_k]$ if φ holds in A with u_i assigned to a_i for each $i \in [k]$. We write $\varphi[A, \alpha; \bar{u}]$ for the set of all tuples $\bar{a} \in A^{\bar{u}}$ with $(A, \alpha[\bar{a}/\bar{u}]) \models \varphi$. For a formula $\varphi(\bar{u})$ (with $\mathrm{free}(\varphi) \subseteq \tilde{u}$) we also denote $\varphi[A, \alpha; \bar{u}]$ by $\varphi[A; \bar{u}]$, and for a formula $\varphi(\bar{v}, \bar{u})$ and $\bar{a} \in A^{\bar{v}}$, we denote $\varphi[A, \alpha[\bar{a}/\bar{v}]; \bar{u}]$ also by $\varphi[A, \bar{a}; \bar{u}]$.

FO+C is obtained by extending FO with the following formula formation rules:

- $\phi := p \leq q$ is a formula if p, q are number variables. We let $\mathrm{free}(\phi) := \{p, q\}$.
- $\phi' := \#\bar{u}\,\psi = \bar{p}$ is a formula if ψ is a formula, $\bar{u}$ is a tuple of variables and $\bar{p}$ a tuple of number variables. We let $\mathrm{free}(\phi') := (\mathrm{free}(\psi) \setminus \tilde{u}) \cup \tilde{p}$.

To define the semantics, let A be a structure and α be an assignment. We let

- $(A, \alpha) \models p \leq q$ iff $\alpha(p) \leq \alpha(q)$,
- $(A, \alpha) \models \#\bar{u}\,\psi = \bar{p}$ iff $|\psi[A, \alpha; \bar{u}]| = \langle \alpha(\bar{p}) \rangle_A$,

where for tuples $\bar{n} = (n_1, \dots, n_k) \in N(A)^k$ we let $\langle \bar{n} \rangle_A$ be the number

$$\langle \bar{n} \rangle_A \;:=\; \sum_{i=1}^{k} n_i \cdot (|U(A)| + 1)^{i-1}.$$

Symmetric transitive closure logic (with counting) STC*(*+C*)* is an extension of FO(+C) with stc-operators. The set of all STC(+C)-formulas is obtained by extending the formula formation rules of FO(+C) by the following rule:

- $\phi := [\mathrm{stc}_{\bar{u},\bar{v}}\,\psi](\bar{u}', \bar{v}')$ is a formula if ψ is a formula and $\bar{u}, \bar{v}, \bar{u}', \bar{v}'$ are compatible tuples of structure (and number) variables. We let $\mathrm{free}(\phi) := \tilde{u}' \cup \tilde{v}' \cup \big(\mathrm{free}(\psi) \setminus (\tilde{u} \cup \tilde{v})\big)$.

Let A be a structure and α be an assignment. We let

- $(A, \alpha) \models [\mathrm{stc}_{\bar{u},\bar{v}}\,\psi](\bar{u}', \bar{v}')$ iff $(\alpha(\bar{u}'), \alpha(\bar{v}'))$ is contained in the symmetric transitive closure of $\psi[A, \alpha; \bar{u}, \bar{v}]$.

$\mathsf{LREC}_=$ is an extension of FO+C with lrec-operators, which allow a limited form of recursion. The lrec-operator controls the depth of the recursion by a "resource term". It thereby makes sure that the recursive definition can be evaluated in logarithmic space. A detailed introduction of $\mathsf{LREC}_=$ can be found in [GGHL12]. Note that we only use previous results about $\mathsf{LREC}_=$ and do not present any formulas using lrec-operators in this paper. We obtain $\mathsf{LREC}_=$ by extending the formula formation rules of FO+C by the following rule:

- $\phi := [\mathrm{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_=,\ \varphi_{\mathtt{E}},\ \varphi_{\mathtt{C}}](\bar{w}, \bar{r})$ is a formula if $\varphi_=$, $\varphi_{\mathtt{E}}$ and $\varphi_{\mathtt{C}}$ are formulas, $\bar{u}, \bar{v}, \bar{w}$ are compatible tuples of variables and $\bar{p}, \bar{r}$ are non-empty tuples of number variables.
  We let $\mathrm{free}(\phi) := \big(\mathrm{free}(\varphi_=) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\mathrm{free}(\varphi_{\mathtt{E}}) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\mathrm{free}(\varphi_{\mathtt{C}}) \setminus (\tilde{u} \cup \tilde{p})\big) \cup \tilde{w} \cup \tilde{r}$.

Let A be a structure and α be an assignment. We let

- $(A, \alpha) \models [\mathrm{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_=,\ \varphi_{\mathtt{E}},\ \varphi_{\mathtt{C}}](\bar{w}, \bar{r})$ iff $\big(\alpha(\bar{w})/_\sim, \langle \alpha(\bar{r}) \rangle_A\big) \in X$,

where X and $\sim$ are defined as follows: Let $\mathtt{V}_0 := A^{\bar{u}}$ and $\mathtt{E}_0 := \varphi_{\mathtt{E}}[A, \alpha; \bar{u}, \bar{v}] \cap (\mathtt{V}_0)^2$. We define $\sim$ to be the reflexive, symmetric, transitive closure of the binary relation $\varphi_=[A, \alpha; \bar{u}, \bar{v}] \cap (\mathtt{V}_0)^2$. Now consider the graph $\mathtt{G} = (\mathtt{V}, \mathtt{E})$ with $\mathtt{V} := \mathtt{V}_0/_\sim$ and $\mathtt{E} := \mathtt{E}_0/_\sim$. To every $\bar{a}/_\sim \in \mathtt{V}$ we assign the set $\mathtt{C}(\bar{a}/_\sim) := \{\langle \bar{n} \rangle_A \mid \text{there is an } \bar{a}' \in \bar{a}/_\sim \text{ with } \bar{n} \in \varphi_{\mathtt{C}}[A, \alpha[\bar{a}'/\bar{u}]; \bar{p}]\}$ of numbers.

Let $\bar{a}/_{\sim}\mathtt{E} := \{\bar{b}/_{\sim} \in \mathtt{V} \mid (\bar{a}/_{\sim}, \bar{b}/_{\sim}) \in \mathtt{E}\}$ and $\mathtt{E}\bar{b}/_{\sim} := \{\bar{a}/_{\sim} \in \mathtt{V} \mid (\bar{a}/_{\sim}, \bar{b}/_{\sim}) \in \mathtt{E}\}$. Then, for all $\bar{a}/_{\sim} \in \mathtt{V}$ and $\ell \in \mathbb{N}$,

$$(\bar{a}/_{\sim}, \ell) \in X \;:\Longleftrightarrow\; \ell > 0 \text{ and } \left|\left\{\bar{b}/_{\sim} \in \bar{a}/_{\sim}\mathtt{E} \;\middle|\; \left(\bar{b}/_{\sim}, \left\lfloor\frac{\ell - 1}{|\mathtt{E}\bar{b}/_{\sim}|}\right\rfloor\right) \in X\right\}\right| \in \mathtt{C}(\bar{a}/_{\sim}).$$

$\mathsf{LREC}_=$ semantically contains $\mathsf{STC{+}C}$ [GGHL12]. Note that simple arithmetics like addition and multiplication are definable in $\mathsf{STC{+}C}$, and therefore, in $\mathsf{LREC}_=$. Like $\mathsf{STC{+}C}$-formulas [Rei05], $\mathsf{LREC}_=$-formulas [GGHL12] can be evaluated in logarithmic space.

2.4. **Transductions.** Transductions (also known as *syntactical interpretations*) define certain structures within other structures. Detailed introductions with a lot of examples can be found in [Gro13, Gru17b]. In the following we briefly introduce transductions, consider compositions of tranductions, and present the new notion of counting transductions.

Definition 2.1 (Parameterized Transduction)**.** Let τ_1, τ_2 be vocabularies, and let L be a logic that extends FO.

(1) A *parameterized* $\mathsf{L}[\tau_1, \tau_2]$*-transduction* is a tuple
$$\Theta(\bar{x}) \;=\; \Big(\theta_{\mathrm{dom}}(\bar{x}), \theta_U(\bar{x}, \bar{u}), \theta_{\approx}(\bar{x}, \bar{u}, \bar{u}'), \big(\theta_R(\bar{x}, \bar{u}_{R,1}, \ldots, \bar{u}_{R,\mathrm{ar}(R)})\big)_{R \in \tau_2}\Big)$$
of $\mathsf{L}[\tau_1]$-formulas, where $\bar{x}$ is a tuple of structure variables, and $\bar{u}, \bar{u}'$ and $\bar{u}_{R,i}$ for every $R \in \tau_2$ and $i \in [\mathrm{ar}(R)]$ are compatible tuples of variables.

(2) The *domain* of $\Theta(\bar{x})$ is the class $\mathrm{Dom}(\Theta(\bar{x}))$ of all pairs $(A, \bar{p})$ such that $A \models \theta_{\mathrm{dom}}[\bar{p}]$, $\theta_U[A, \bar{p}; \bar{u}]$ is not empty and $\theta_{\approx}[A, \bar{p}; \bar{u}, \bar{u}']$ is an equivalence relation, where A is a τ_1-structure and $\bar{p} \in A^{\bar{x}}$. The elements in $\bar{p}$ are called *parameters*.

(3) Let $(A, \bar{p})$ be in the domain of $\Theta(\bar{x})$, and let us denote $\theta_{\approx}[A, \bar{p}; \bar{u}, \bar{u}']$ by $\approx$. We define a τ_2-structure $\Theta[A, \bar{p}]$ as follows. We let
$$U(\Theta[A, \bar{p}]) \;:=\; \theta_U[A, \bar{p}; \bar{u}]/_{\approx}$$
be the universe of $\Theta[A, \bar{p}]$. Further, for each $R \in \tau_2$, we let
$$R(\Theta[A, \bar{p}]) := \Big(\theta_R[A, \bar{p}; \bar{u}_{R,1}, \ldots, \bar{u}_{R,\mathrm{ar}(R)}] \cap \theta_U[A, \bar{p}; \bar{u}]^{\mathrm{ar}(R)}\Big)\Big/_{\approx}.$$

A parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction defines a parameterized mapping from τ_1-structures into τ_2-structures via $\mathsf{L}[\tau_1]$-formulas.[3] If $\theta_{\mathrm{dom}} := \top$ or $\theta_{\approx} := u_1 = u_1' \wedge \cdots \wedge u_k = u_k'$, we omit the respective formula in the presentation of the transduction. A parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction $\Theta(\bar{x})$ is an $\mathsf{L}[\tau_1, \tau_2]$*-transduction* if $\bar{x}$ is the empty tuple. Let $\bar{x}$ be the empty tuple. For simplicity, we denote a transduction $\Theta(\bar{x})$ by Θ, and we write $A \in \mathrm{Dom}(\Theta)$ if $(A, \bar{x})$ is contained in the domain of Θ.

An important property of $\mathsf{L}[\tau_1, \tau_2]$-transductions is that, for suitable logics L, they allow to *pull back* $\mathsf{L}[\tau_2]$-formulas, which means that for each $\mathsf{L}[\tau_2]$-formula there exists an $\mathsf{L}[\tau_1]$-formula that expresses essentially the same. A logic L is *closed under (parameterized)* L*-transductions* if for all vocabularies τ_1, τ_2 each (parameterized) $\mathsf{L}[\tau_1, \tau_2]$-transduction allows to pull back $\mathsf{L}[\tau_2]$-formulas.

Let $\mathcal{L}$ be the following set of logics:
$$\mathcal{L} := \{\mathsf{FO}, \mathsf{FO{+}C}, \mathsf{STC}, \mathsf{STC{+}C}, \mathsf{LREC}_=\}.$$

[3] In Section 4.2, for example, we will define a parameterized STC-transduction that maps trees to directed trees. It uses a leaf r of a tree T as a parameter to root the tree T at r.

Each logic $\mathsf{L} \in \mathcal{L}$ is closed under L-transductions. Precisely, this means that:

Proposition 2.2 [EF99, GGHL12, Gru17b]**.** *Let τ_1, τ_2 be vocabularies and $\mathsf{L} \in \mathcal{L}$. Let $\Theta(\bar{x})$ be a parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction, where ℓ-tuple $\bar{u}$ is the tuple of domain variables. Further, let $\psi(x_1, \dots, x_\kappa, p_1, \dots, p_\lambda)$ be an $\mathsf{L}[\tau_2]$-formula where $x_1, \dots, x_\kappa$ are structure variables and $p_1, \dots, p_\lambda$ are number variables. Then there exists an $\mathsf{L}[\tau_1]$-formula $\psi^{-\Theta}(\bar{x}, \bar{u}_1, \dots, \bar{u}_\kappa, \bar{q}_1, \dots, \bar{q}_\lambda)$, where $\bar{u}_1, \dots, \bar{u}_\kappa$ are compatible with $\bar{u}$ and $\bar{q}_1, \dots, \bar{q}_\lambda$ are ℓ-tuples of number variables, such that for all $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$, all $\bar{a}_1, \dots, \bar{a}_\kappa \in A^{\bar{u}}$ and all $\bar{n}_1, \dots, \bar{n}_\lambda \in N(A)^\ell$,*

$$A \models \psi^{-\Theta}[\bar{p}, \bar{a}_1, \dots, \bar{a}_\kappa, \bar{n}_1, \dots, \bar{n}_\lambda] \iff \begin{array}{l} \bar{a}_1/_{\approx}, \dots, \bar{a}_\kappa/_{\approx} \in U(\Theta[A, \bar{p}]), \\ \langle \bar{n}_1 \rangle_A, \dots, \langle \bar{n}_\lambda \rangle_A \in N(\Theta[A, \bar{p}]) \text{ and} \\ \Theta[A, \bar{p}] \models \psi\big[\bar{a}_1/_{\approx}, \dots, \bar{a}_\kappa/_{\approx}, \langle \bar{n}_1 \rangle_A, \dots, \langle \bar{n}_\lambda \rangle_A\big], \end{array}$$

where $\approx$ is the equivalence relation $\theta_\approx[A, \bar{p}; \bar{u}, \bar{u}']$ on $A^{\bar{u}}$.

The following proposition shows that for each logic $\mathsf{L} \in \mathcal{L}$, the composition of a parameterized L-transduction and an L-transduction is again a parameterized L-transduction. Note that this is a consequence of Proposition 2.2.

Proposition 2.3 [Gru17b]**.** *Let τ_1, τ_2 and τ_3 be vocabularies and let $\mathsf{L} \in \mathcal{L}$. Let $\Theta_1\big(\bar{x}\big)$ be a parameterized $\mathsf{L}[\tau_1, \tau_2]$-transduction and Θ_2 be an $\mathsf{L}[\tau_2, \tau_3]$-transduction. Then there exists a parameterized $\mathsf{L}[\tau_1, \tau_3]$-transduction $\Theta\big(\bar{x}\big)$ such that for all τ_1-structures A and all $\bar{p} \in A^{\bar{x}}$,*

$$\big(A, \bar{p}\big) \in \mathrm{Dom}\big(\Theta\big(\bar{x}\big)\big) \iff \big(A, \bar{p}\big) \in \mathrm{Dom}\big(\Theta_1\big(\bar{x}\big)\big) \;\; \text{and} \;\; \Theta_1\big[A, \bar{p}\big] \in \mathrm{Dom}\big(\Theta_2\big),$$

and for all $\big(A, \bar{p}\big) \in \mathrm{Dom}\big(\Theta\big(\bar{x}\big)\big)$,

$$\Theta\big[A, \bar{p}\big] \cong \Theta_2\big[\Theta_1\big[A, \bar{p}\big]\big].$$

In the following we introduce the new notion of parameterized counting transductions for STC+C. The universe of the structure $\Theta^{\#}[A, \bar{p}]$ defined by a parameterized counting transduction $\Theta^{\#}(\bar{x})$ always also includes the number sort $N(A)$ of A, for all structures A and tuples $\bar{p}$ of parameters from the domain of $\Theta^{\#}(\bar{x})$. More precisely, the universe of $\Theta^{\#}[A, \bar{p}]$ contains all equivalence classes $\{n\}$ where $n \in N(A)$ and all equivalence classes that the universe of $\Theta^{\#}[A, \bar{p}]$ would contain if we interpreted the parameterized counting transduction $\Theta^{\#}(\bar{x})$ as a parameterized transduction. Parameterized counting transductions are as powerful as parameterized transductions. Presenting a parameterized counting transduction instead of a parameterized transduction will contribute to a clearer presentation.

Definition 2.4 (Parameterized Counting Transduction)**.** Let τ_1, τ_2 be vocabularies.

(1) A *parameterized* STC+$\mathsf{C}[\tau_1, \tau_2]$*-counting transduction* is a tuple

$$\Theta^{\#}(\bar{x}) \;=\; \Big(\theta^{\#}_{\mathrm{dom}}(\bar{x}), \theta^{\#}_{U}(\bar{x}, \bar{u}), \theta^{\#}_{\approx}(\bar{x}, \bar{u}, \bar{u}'), \big(\theta^{\#}_{R}(\bar{x}, \bar{u}_{R,1}, \dots, \bar{u}_{R,\mathrm{ar}(R)})\big)_{R \in \tau_2}\Big)$$

of STC+$\mathsf{C}[\tau_1]$-formulas, where $\bar{x}$ is a tuple of structure variables, $\bar{u}, \bar{u}'$ are compatible tuples of variables but not tuples of number variables of length 1,[4] and for every $R \in \tau_2$ and $i \in [\mathrm{ar}(R)]$, $\bar{u}_{R,i}$ is a tuple of variables that is compatible to $\bar{u}$ or a tuple of number variables of length 1.

[4] We do not allow $\bar{u}, \bar{u}'$ to be tuples of number variables of length 1, as the equivalence classes $\{n\}$ for $n \in N(A)$ are always added to the universe of $\Theta^{\#}[A, \bar{p}]$. This will become more clear with the definition of the universe $U(\Theta^{\#}[A, \bar{p}])$ in (3).

(2) The *domain* of $\Theta^{\#}(\bar{x})$ is the class $\mathrm{Dom}(\Theta^{\#}(\bar{x}))$ of all pairs $(A, \bar{p})$ such that $A \models \theta^{\#}_{\mathrm{dom}}[\bar{p}]$ and $\theta^{\#}_{\approx}[A, \bar{p}; \bar{u}, \bar{u}']$ is an equivalence relation, where A is a τ_1-structure and $\bar{p} \in A^{\bar{x}}$.

(3) Let $(A, \bar{p})$ be in the domain of counting transduction $\Theta^{\#}(\bar{x})$ and let us denote the equivalence relation $\theta^{\#}_{\approx}[A, \bar{p}; \bar{u}, \bar{u}'] \cup \{(n, n) \mid n \in N(A)\}$ by $\approx$. We define a τ_2-structure $\Theta^{\#}[A, \bar{p}]$ as follows. We let

$$U(\Theta^{\#}[A, \bar{p}]) := \big(\theta^{\#}_{U}[A, \bar{p}; \bar{u}] \mathbin{\dot\cup} N(A)\big)/_{\approx}$$

be the universe of $\Theta^{\#}[A, \bar{p}]$. Further, for each $R \in \tau_2$, we let

$$R(\Theta^{\#}[A, \bar{p}]) := \Big(\theta^{\#}_{R}[A, \bar{p}; \bar{u}_{R,1}, \ldots, \bar{u}_{R,\mathrm{ar}(R)}] \cap \big(\theta^{\#}_{U}[A, \bar{p}; \bar{u}] \mathbin{\dot\cup} N(A)\big)^{\mathrm{ar}(R)}\Big)\Big/_{\approx}.$$

Proposition 2.5 [Gru17b]**.** *Let $\Theta^{\#}(\bar{x})$ be a parameterized $\mathsf{STC{+}C}[\tau_1, \tau_2]$-counting transduction. Then there exists a parameterized $\mathsf{STC{+}C}[\tau_1, \tau_2]$-transduction $\Theta(\bar{x})$ such that*

- $\mathrm{Dom}(\Theta(\bar{x})) = \mathrm{Dom}(\Theta^{\#}(\bar{x}))$ *and*
- $\Theta[A, \bar{p}] \cong \Theta^{\#}[A, \bar{p}]$ *for all $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$.*

2.5. **Canonization.** In this section we introduce ordered structures, (definable) canonization and the capturing of the complexity class LOGSPACE.

Let τ be a vocabulary with $\leq \notin \tau$. A $\tau \cup \{\leq\}$-structure A' is *ordered* if the relation symbol $\leq$ is interpreted as a linear order on the universe of A'. Let A be a τ-structure. An ordered $\tau \cup \{\leq\}$-structure A' is an *ordered copy* of A if $A'|_{\tau} \cong A$. Let $\mathcal{C}$ be a class of τ-structures. A mapping f is a *canonization mapping* of $\mathcal{C}$ if it assigns every structure $A \in \mathcal{C}$ to an ordered copy $f(A) = (A_f, \leq_f)$ of A such that for all structures $A, B \in \mathcal{C}$ we have $f(A) \cong f(B)$ if $A \cong B$. We call the ordered structure $f(A)$ the *canon* of A.

Let L be a logic that extends FO. Let $\Theta(\bar{x})$ be a parameterized $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction, where $\bar{x}$ is a tuple of structure variables. We say $\Theta(\bar{x})$ *canonizes* a τ-structure A if there exists a tuple $\bar{p} \in A^{\bar{x}}$ such that $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$, and for all tuples $\bar{p} \in A^{\bar{x}}$ with $(A, \bar{p}) \in \mathrm{Dom}(\Theta(\bar{x}))$, the $\tau \cup \{\leq\}$-structure $\Theta[A, \bar{p}]$ is an ordered copy of A. Note that if the tuple $\bar{x}$ of parameter variables is the empty tuple, $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction Θ canonizes a τ-structure A if $A \in \mathrm{Dom}(\Theta)$ and the $\tau \cup \{\leq\}$-structure $\Theta[A]$ is an ordered copy of A. A *(parameterized)* L*-canonization* of a class $\mathcal{C}$ of τ-structures is a (parameterized) $\mathsf{L}[\tau, \tau \cup \{\leq\}]$-transduction that canonizes all $A \in \mathcal{C}$. A class $\mathcal{C}$ of τ-structures *admits* L*-definable canonization* if $\mathcal{C}$ has a (parameterized) L-canonization.

The following proposition and theorem are essential for proving that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization in Section 5.

Proposition 2.6 [Gro13][5]**.** *Let $\mathcal{C}$ be a class of graphs, and $\mathcal{C}_{\mathrm{con}}$ be the class of all connected components of the graphs in $\mathcal{C}$. If $\mathcal{C}_{\mathrm{con}}$ admits $\mathsf{LREC}_=$-definable canonization, then $\mathcal{C}$ does as well.*

[5] In [Gro13, Corollary 3.3.21] Proposition 2.6 is only shown for $\mathsf{IFP{+}C}$. The proof of Corollary 3.3.21 uses Lemma 3.3.18, the Transduction Lemma, and that connectivity and simple arithmetics are definable. As $\mathsf{LREC}_=$ is closed under parameterized $\mathsf{LREC}_=$-transductions, the Transduction Lemma also holds for $\mathsf{LREC}_=$ [GGHL12]. Connectivity and all arithmetics (e.g., addition, multiplication and Fact 3.3.14) that are necessary to show Lemma 3.3.18 and Corollary 3.3.21 can also be defined in $\mathsf{LREC}_=$. Further, Lemma 3.3.12 and 3.3.17, which are used to prove Lemma 3.3.18 can be shown by pulling back simple FO-formulas under $\mathsf{LREC}_=$-transductions. Hence, Corollary 3.3.21 also holds for $\mathsf{LREC}_=$.

Theorem 2.7 [GGHL12, Gru17b][6]**.** *The class of* LO*-colored directed trees admits* $\mathsf{LREC}_=$*-definable canonization.*

We can use definable canonization of a graph class to prove that LOGSPACE is captured on this graph class. Let L be a logic and $\mathcal{C}$ be a graph class. L *captures* LOGSPACE *on* $\mathcal{C}$ if for each class $\mathcal{D} \subseteq \mathcal{C}$, there exists an L-sentence defining $\mathcal{D}$ if and only if $\mathcal{D}$ is LOGSPACE-decidable. A precise definition of what it means that a logic *(effectively strongly) captures* a complexity class can be found in [EF99, Chapter 11]. A fundamental result was shown by Immerman:

Theorem 2.8 [Imm87][7]**.** DTC *captures* LOGSPACE *on the class of all ordered graphs.*

Deterministic transitive closure logic DTC is a logic that is contained in $\mathsf{LREC}_=$ [GGHL12]. Since $\mathsf{LREC}_=$-formulas can be evaluated in logarithmic space [GGHL12], we obtain the following corollary:

Corollary 2.9. $\mathsf{LREC}_=$ *captures* LOGSPACE *on the class of all ordered graphs.*

Let us suppose there exists a parameterized $\mathsf{LREC}_=$-canonization of a graph class $\mathcal{C}$. Since $\mathsf{LREC}_=$ captures LOGSPACE on the class of all ordered graphs and we can pull back each $\mathsf{LREC}_=$-sentence that defines a logarithmic-space property on ordered graphs under this canonization, the capturing result transfers from ordered graphs to the class $\mathcal{C}$.

Proposition 2.10. *Let $\mathcal{C}$ be a class of graphs. If $\mathcal{C}$ admits $\mathsf{LREC}_=$-definable canonization, then $\mathsf{LREC}_=$ captures* LOGSPACE *on $\mathcal{C}$.*

3. Clique Trees and their Structure

Clique trees of connected chordal claw-free graphs play an important role in our canonization of the class of chordal claw-free graphs. Thus, we analyze the structure of clique trees of connected chordal claw-free graphs in this section.

First we introduce clique trees of chordal graphs. Then we show that chordal claw-free graphs are intersection graphs of paths of a tree. We use this property to prove that each connected chordal claw-free graph has a unique clique tree. Finally, we introduce two different types of max cliques in a clique tree, star cliques and fork cliques, and show that each max clique of a connected chordal claw-free graph is of one of these types if its degree in the clique tree is at least 3.

3.1. **Clique Trees of Chordal Graphs.**

Chordal graphs are precisely the intersection graphs of subtrees of a tree. A clique tree of a chordal graph G specifies a minimal representation of G as such an intersection graph. Clique trees were introduced independently by Buneman [Bun74], Gavril [Gav74] and Walter [Wal72]. A detailed introduction of chordal graphs and their clique trees can be found in [BP93].

[6] It is shown in [GGHL12, Remark 4.8], and in more detail in [Gru17b, Section 8.4] that the class of all colored directed trees that have a linear order on the colors admits LREC-definable canonization. This can easily be extended to LO-colored directed trees since an LO-colored directed tree is a special kind of colored directed tree that has a linear order on its colors. LREC is contained in $\mathsf{LREC}_=$ [GGHL12].

[7] Immerman proved this capturing result not only for the class of ordered graphs but for the class of ordered structures.

Let G be a chordal graph, and let $\mathcal{M}$ be the set of max cliques of G. Further, let $\mathcal{M}_v$ be the set of all max cliques in $\mathcal{M}$ that contain a vertex v of G. A *clique tree* of G is a tree $T = (\mathcal{M}, \mathcal{E})$ whose vertex set is the set $\mathcal{M}$ of all max cliques where for all $v \in V$ the induced subgraph $T[\mathcal{M}_v]$ is connected. Hence, for each $v \in V$ the induced subgraph $T[\mathcal{M}_v]$ is a subtree of T. Then G is the intersection graph of the subtrees $T[\mathcal{M}_v]$ of T where $v \in V$. An example of a clique tree of a chordal graph is shown in Figure 1.

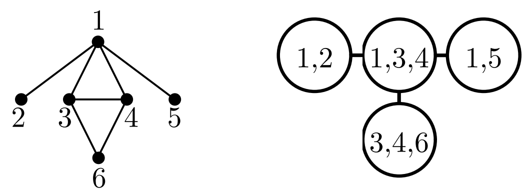


Figure 1. A chordal graph and a clique tree of the graph

Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal graph G. It is easy to see that the clique tree T satisfies the *clique intersection property*: Let $M_1, M_2, M_3 \in \mathcal{M}$ be vertices of the tree T. If M_2 is on the path from M_1 to M_3, then $M_1 \cap M_3 \subseteq M_2$.

3.2. Intersection-Graph Representation of Chordal Claw-Free Graphs.

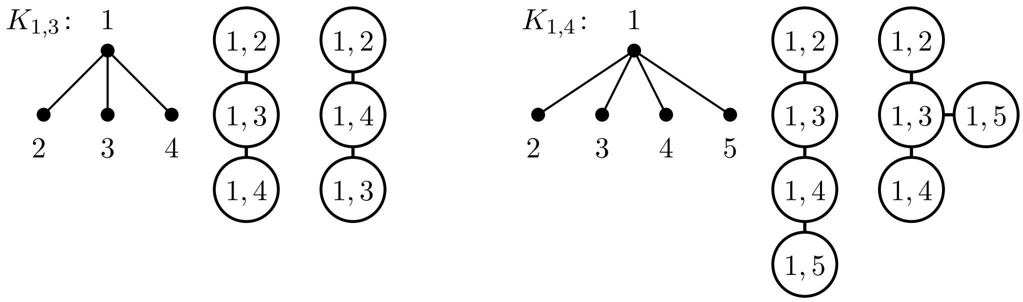
In the following we consider the class CCF, i.e., the class of chordal claw-free graphs. For each vertex v of a chordal claw-free graph, we prove that the set of max cliques $\mathcal{M}_v$ induces a path in each clique tree. Consequently, chordal claw-free graphs are intersection graphs of paths of a tree. Note that not all intersection graphs of paths of a tree are claw-free (see Figure 1).

Lemma 3.1. *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph $G = (V, E)$. Then for all $v \in V$ the induced subtree $T[\mathcal{M}_v]$ is a path in T.*

Proof. Let $G = (V, E) \in$ CCF and let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Let us assume there exists a vertex $v \in V$ such that the graph $T[\mathcal{M}_v]$ is not a path in T. As $T[\mathcal{M}_v]$ is a subtree of T, there exists a max clique $B \in \mathcal{M}_v$ such that B has degree at least 3. Let $A_1, A_2, A_3 \in \mathcal{M}_v$ be three distinct neighbors of B in $T[\mathcal{M}_v]$. Since A_i and B are distinct max cliques, there exists a vertex $a_i \in A_i \setminus B$, and for each $i \in [3]$, we have $A_i \in \mathcal{M}_{a_i}$, $B \notin \mathcal{M}_{a_i}$ and $T[\mathcal{M}_{a_i}]$ is connected. As T is a tree, A_1, A_2, and A_3 are all in different connected components of $T[\mathcal{M} \setminus \{B\}]$. Therefore, $\mathcal{M}_{a_i} \cap \mathcal{M}_{a_{i'}} = \emptyset$ for all $i, i' \in [3]$ with $i \neq i'$. Now, $\{v, a_1, a_2, a_3\}$ induces a claw in G, which contradicts G being claw-free: For all $i \in [3]$, there is an edge between v and a_i, because $v, a_i \in A_i$. To show that vertices a_i and $a_{i'}$ are not adjacent for $i \neq i'$, let us assume the opposite. If a_i and $a_{i'}$ are adjacent, then there exists a max clique M containing a_i and $a_{i'}$. Thus, $\mathcal{M}_{a_i} \cap \mathcal{M}_{a_{i'}} \neq \emptyset$, a contradiction. □

3.3. Uniqueness of the Clique Tree for Connected Chordal Claw-Free Graphs.

The following lemmas help us to show in Corollary 3.6 that the clique tree of a connected chordal claw-free graph is unique. Notice, that this is a property that generally does not hold for unconnected graphs. Given an unconnected chordal (claw-free) graph, we can connect the clique trees for the connected components in an arbitrary way to obtain a clique tree of the entire graph. Further, connected chordal graphs in general also do not have a unique clique tree. For example, the claw is a connected chordal graph having multiple clique trees (see Figure 2A), and the $K_{1,4}$ is a connected chordal graph where the clique trees are not even isomorphic (see Figure 2B).



(A) The $K_{1,3}$ and two clique trees (B) The $K_{1,4}$ and two non-isomorphic clique trees

FIGURE 2. Connected chordal graphs where the clique tree is not unique

Lemma 3.2. *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph $G = (V, E)$. Further, let $v \in V$, and let A_1, A_2, A_3 be distinct max cliques in $\mathcal{M}_v$. Then A_2 lies between A_1 and A_3 on the path $T[\mathcal{M}_v]$ if and only if $A_2 \subseteq A_1 \cup A_3$.*

Proof. Let $G = (V, E) \in$ CCF and $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Further, let $v \in V$, and let $A_1, A_2, A_3 \in \mathcal{M}_v$ be distinct max cliques. First, suppose $A_2 \subseteq A_1 \cup A_3$, and let us assume that, w.l.o.g., A_1 lies between A_2 and A_3. Then $A_2 \cap A_3 \subseteq A_1$ according to the clique intersection property. Further, $A_2 \subseteq A_1 \cup A_3$ implies that $A_2 \setminus A_3 \subseteq A_1$. It follows that $A_2 \subseteq A_1$, which is a contradiction to A_1 and A_2 being distinct max cliques.

Now let max clique A_2 lie between A_1 and A_3 on the path $T[\mathcal{M}_v]$, and let us assume that there exists a vertex $a_2 \in A_2 \setminus (A_1 \cup A_3)$. Let $P = B_1, \dots, B_l$ be the path $T[\mathcal{M}_v]$ (Lemma 3.1). W.l.o.g., assume that $A_i = B_{j_i}$ for all $i \in [3]$ where $j_1, j_2, j_3 \in [l]$ with $j_1 < j_2 < j_3$. Further, let $A'_1 := B_{j_1+1}$ and $A'_3 := B_{j_3-1}$, and let $a_1 \in A_1 \setminus A'_1$ and $a_3 \in A_3 \setminus A'_3$. Similarly to the proof of Lemma 3.1, we obtain that $\{v, a_1, a_2, a_3\}$ induces a claw in G, a contradiction. □

Corollary 3.3. *For all distinct vertices $v, w \in V$, the graph $T[\mathcal{M}_v \setminus \mathcal{M}_w]$ is connected.*[8]

Proof. Let $v, w \in V$ be distinct vertices. Let $P = A_1, \dots, A_l$ be the path $T[\mathcal{M}_v]$, and let us assume $T[\mathcal{M}_v \setminus \mathcal{M}_w]$ is not connected. Then there exist $i, j, k \in [l]$ with $i < j < k$ such that $A_i, A_k \in \mathcal{M}_v \setminus \mathcal{M}_w$ and $A_j \in \mathcal{M}_w$. By Lemma 3.2 we have $A_j \subseteq A_i \cup A_k$. Thus, vertex $w \in A_j$ is also contained in A_i or A_k, a contradiction. □

Lemma 3.4. *Let $T_1 = (\mathcal{M}, \mathcal{E}_1)$ and $T_2 = (\mathcal{M}, \mathcal{E}_2)$ be clique trees of a chordal claw-free graph $G = (V, E)$. Then for every $v \in V$ we have $T_1[\mathcal{M}_v] = T_2[\mathcal{M}_v]$.*

Proof. Let $G = (V, E) \in$ CCF and let $T_1 = (\mathcal{M}, \mathcal{E}_1)$ and $T_2 = (\mathcal{M}, \mathcal{E}_2)$ be clique trees of G. Let $v \in V$. According to Lemma 3.1, $T_1[\mathcal{M}_v]$ and $T_2[\mathcal{M}_v]$ are paths in T_1 and T_2, respectively. Let us assume there exist distinct max cliques $A, B \in \mathcal{M}_v$ such that, A, B are adjacent in $T_1[\mathcal{M}_v]$ but not adjacent in $T_2[\mathcal{M}_v]$. As A and B are not adjacent in $T_2[\mathcal{M}_v]$, there exists a max clique $C \in \mathcal{M}_v$ that lies between A and B on the path $T_2[\mathcal{M}_v]$. Thus, $A \cap B \subseteq C$ according to the clique intersection property. Since max cliques A and B are adjacent in $T_1[\mathcal{M}_v]$, either A lies between B and C, or B lies between A and C on the path $T_1[\mathcal{M}_v]$.

[8]We define the empty graph as connected.

W.l.o.g., suppose that A lies between B and C on the path $T_1[\mathcal{M}_v]$. Then $A \subseteq B \cup C$ by Lemma 3.2. Thus, we have $A \setminus B \subseteq C$. Since $A \cap B \subseteq C$, this yields that $A \subseteq C$, which is a contradiction to A and C being distinct max cliques. □

Lemma 3.5. *Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a connected chordal graph $G = (V, E)$. Then*

$$T = \bigcup_{v \in V} T[\mathcal{M}_v].$$

Proof. Let $G = (V, E)$ be a connected chordal graph and $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of G. Clearly, the graphs T and $T' := \bigcup_{v \in V} T[\mathcal{M}_v]$ have the same vertex set, and T' is a subgraph of the tree T. In order to prove that $T = T'$, we show that T' is connected.

For all vertices $v \in V$, the graph $T'[\mathcal{M}_v]$ is connected because $T[\mathcal{M}_v]$ is connected. For each edge $\{u, v\} \in E$ of the graph G, there exists a max clique that contains u and v, and therefore, we have $\mathcal{M}_u \cap \mathcal{M}_v \neq \emptyset$. Hence, $T'[\mathcal{M}_u \cup \mathcal{M}_v]$ is connected for every edge $\{u, v\} \in E$. Since G is connected, it follows that $T'[\bigcup_{v \in V} \mathcal{M}_v]$ is connected. Clearly, $\bigcup_{v \in V} \mathcal{M}_v = \mathcal{M}$. Consequently, the graph T' is connected. □

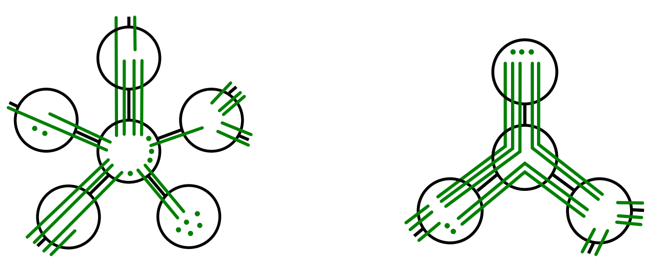
As a direct consequence of Lemma 3.4 and Lemma 3.5 we obtain the following corollary. It follows that each connected chordal claw-free graph has a unique clique tree.

Corollary 3.6. *Let T_1 and T_2 be clique trees of a connected chordal claw-free graph G. Then $T_1 = T_2$.*

3.4. **Star Cliques and Fork Cliques.** In the following let $G = (V, E)$ be a connected chordal claw-free graph and let $T_G = (\mathcal{M}, \mathcal{E})$ be its clique tree.

Let B be a max clique of G. If for all $v \in B$ max clique B is an end of path $T_G[\mathcal{M}_v]$, we call B a *star clique*. Thus, B is a star clique if, and only if, every vertex in B is contained in at most one neighbor of B in T_G. A picture of a star clique can be found in Figure 3a. Clearly, every max clique of degree 1, i.e., every leaf, of clique tree T_G is a star clique.

A max clique B of degree 3 is called a *fork clique* if for every $v \in B$ there exist two neighbors A, A' of B with $A \neq A'$ such that $\mathcal{M}_v = \{B, A, A'\}$, and for all neighbors A, A' of B with $A \neq A'$ there exists a vertex $v \in B$ with $\mathcal{M}_v = \{B, A, A'\}$. Figure 3b shows a sketch of a fork clique. Note that two fork cliques cannot be adjacent.



(a) A star clique (b) A fork clique

Figure 3. A star clique and a fork clique. Each picture shows a part of a clique tree T_G. For $v \in V$ each path $T_G[\mathcal{M}_v]$ is depicted as a green line.

The following lemma and corollary provide more information about the structure of the clique tree of a connected chordal claw-free graph.

Lemma 3.7. *Let $B \in \mathcal{M}$. If the degree of B in clique tree T_G is at least 3, then B is a star clique or a fork clique.*

Corollary 3.8. *Let $B \in \mathcal{M}$ be a fork clique. Then every neighbor of B in clique tree T_G is a star clique.*

Proof. Let us assume max clique A is a neighbor of fork clique B, and A is not a star clique. Then the degree of A is at least 2. As A cannot be a fork clique, Lemma 3.7 implies that A has degree 2. Since B is a fork clique, there does not exist a vertex $v \in A$ that is contained in B and the other neighbor of A. Thus, A is a star clique, a contradiction. □

In the remainder of this section we prove Lemma 3.7.

Let P and Q be two paths in T_G. We call $(A', A, \{A_P, A_Q\}) \in V^2 \times \binom{V}{2}$ a *fork* of P and Q, if $P[\{A', A, A_P\}]$ and $Q[\{A', A, A_Q\}]$ are induced subpaths of length 3 of P and Q, respectively, and neither A_P occurs in Q nor A_Q occurs in P. Figure 4 shows a fork of paths P and Q. We say P and Q *fork (in A)* if there exists a fork $(A', A, \{A_P, A_Q\})$ of P and Q.

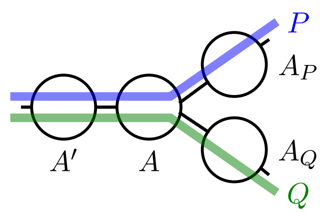


FIGURE 4. A fork of P and Q

Lemma 3.9. *Let $v, w \in V$. If the paths $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ fork, then $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ are paths of length 3.*

Proof. Let $v, w \in V$. Clearly, if $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ fork, then they must be paths of length at least 3. It remains to prove that their length is at most 3. For a contradiction, let us assume the length of $T_G[\mathcal{M}_v]$ is at least 4. Let $(A_1, B, \{A_2, A_2'\})$ be a fork of $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_w]$ where $A_2 \in \mathcal{M}_v \setminus \mathcal{M}_w$ and $A_2' \in \mathcal{M}_w \setminus \mathcal{M}_v$.

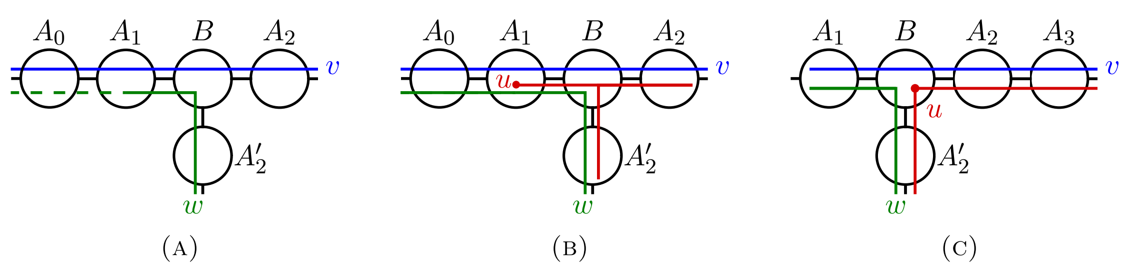


FIGURE 5. Illustrations for the proof of Lemma 3.9

First let us assume there exists a max clique $A_0 \in \mathcal{M}_v$ such that $P = A_0, A_1, B, A_2$ is a subpath of $T_G[\mathcal{M}_v]$ of length 4. According to Corollary 3.3, the graph $T_G[\mathcal{M}_v \setminus \mathcal{M}_w]$ is connected. Thus, we have $A_0 \in \mathcal{M}_w$ (see Figure 5A). Now A_0 and A_1 are distinct max cliques. Therefore, there exists a vertex $u \in A_1 \setminus A_0$. As P is a subpath of $T_G[\mathcal{M}_v]$ and $P' = A_0, A_1, B, A_2'$ is a subpath of $T_G[\mathcal{M}_w]$, vertex u is not only contained in A_1 but also in B, A_2 and A_2' by Lemma 3.2 (see Figure 5B). As a consequence, $T_G[\mathcal{M}_u]$ is not a path, a contradiction to Lemma 3.1.

Next, let us assume there exists a max clique $A_3 \in \mathcal{M}_v$ such that $P = A_1, B, A_2, A_3$ is a subpath of $T_G[\mathcal{M}_v]$ of length 4. Further, $P' = A_1, B, A_2'$ is a subpath of $T_G[\mathcal{M}_w]$. As A_1 and B are max cliques, there exists a vertex $u \in B \setminus A_1$. By Lemma 3.2, vertex u is also contained in A_2, A_3 and A_2' as shown in Figure 5c. Now let us consider the paths $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_u]$. $Q = A_3, A_2, B, A_1$ is a subpath of $T_G[\mathcal{M}_v]$, and $Q' = A_3, A_2, B, A_2'$ is a subpath of $T_G[\mathcal{M}_u]$. Clearly, $(A_2, B, \{A_1, A_2'\})$ is a fork of $T_G[\mathcal{M}_v]$ and $T_G[\mathcal{M}_u]$. According to the previous part of this proof, we obtain a contradiction. □

The max cliques $A_1, A_2, A_3 \in \mathcal{M}$ form a *fork triangle* around a max clique $B \in \mathcal{M}$ if A_1, A_2 and A_3 are distinct neighbors of B and there exist vertices $u, v, w \in V$ such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. We say that max clique $B \in \mathcal{M}$ has a *fork triangle* if there exist max cliques $A_1, A_2, A_3 \in \mathcal{M}$ that form a fork triangle around B. Figure 6 depicts a fork triangle around a max clique B. Clearly, if a max clique B has a fork triangle, then B is a vertex of degree at least 3 in T_G.

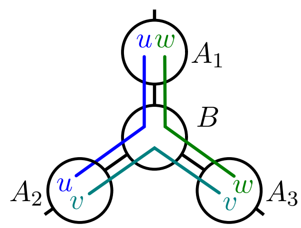


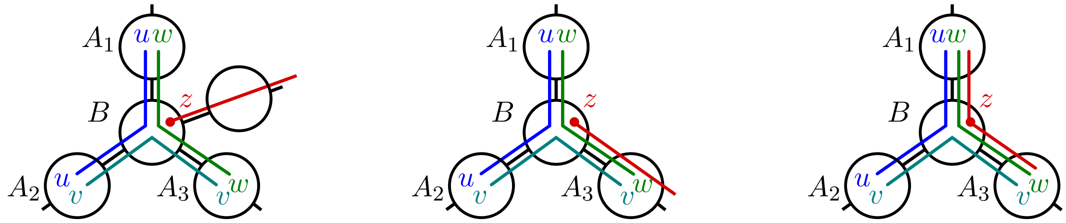
Figure 6. A fork triangle

Lemma 3.10. *Let $v, w \in V$, and let $B \in \mathcal{M}$ be a max clique. If $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ fork in B, then B has a fork triangle.*

Proof. Let $v, w \in V$, let $B \in \mathcal{M}$ be a max clique, and let $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ fork in B. Then $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$ are paths of length 3 by Lemma 3.9. Let $\mathcal{M}_u = \{A_2, B, A_1\}$ and $\mathcal{M}_v = \{A_2, B, A_3\}$ with $A_1 \neq A_3$. Since B and A_2 are max cliques, there exists a vertex $w \in B \setminus A_2$. Now, we can apply Lemma 3.2 to the paths $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_v]$, and obtain that $w \in A_1$ and $w \in A_3$. As $T_G[\mathcal{M}_w]$ and $T_G[\mathcal{M}_u]$ fork, the path $T_G[\mathcal{M}_w]$ must be of length 3 by Lemma 3.9. Thus, $\mathcal{M}_w = \{A_3, B, A_1\}$. Hence, A_1, A_2, A_3 form a fork triangle around B. □

Lemma 3.11. *Let $z \in V$. If max clique $B \in \mathcal{M}_z$ has a fork triangle, then $|\mathcal{M}_z| = 3$ and B is in the middle of path $T_G[\mathcal{M}_z]$.*

Proof. Let $z \in V$, and let $B \in \mathcal{M}_z$ have a fork triangle. Then, there exist $u, v, w \in V$ and distinct neighbor max cliques A_1, A_2, A_3 of B such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. Let $\mathcal{W}$ be the set $\{A_1, A_2, A_3\}$ of max cliques that form a fork triangle around B. Let us consider $|\mathcal{M}_z \cap \mathcal{W}|$. If $|\mathcal{M}_z \cap \mathcal{W}| \leq 1$, then $\mathcal{M}_z$ is a separating set of at least one of the paths $T_G[\mathcal{M}_u]$, $T_G[\mathcal{M}_v]$ or $T_G[\mathcal{M}_w]$ as shown in Figure 7a and 7b, and we have a contradiction to Corollary 3.3. Clearly, we cannot have $|\mathcal{M}_z \cap \mathcal{W}| = 3$, since $T_G[\mathcal{M}_z]$ must be a path. It remains to consider $|\mathcal{M}_z \cap \mathcal{W}| = 2$, which is illustrated in Figure 7c. In this case, $T_G[\mathcal{M}_z]$ forks with one of the paths $T_G[\mathcal{M}_u]$, $T_G[\mathcal{M}_v]$ or $T_G[\mathcal{M}_w]$ in B, and must be of length 3 according to Lemma 3.9. Obviously, B is in the middle of the path $T_G[\mathcal{M}_z]$. □



(A) $|\mathcal{M}_z \cap \mathcal{W}| = 0$ (B) $|\mathcal{M}_z \cap \mathcal{W}| = 1$ (C) $|\mathcal{M}_z \cap \mathcal{W}| = 2$

FIGURE 7. Illustrations for the proof of Lemma 3.11

Lemma 3.12. *If max clique $B \in \mathcal{M}$ has a fork triangle, then the degree of B in T_G is* 3.

Proof. Let $B \in \mathcal{M}$ have a fork triangle. Thus, there exists vertices $u, v, w \in V$ and distinct neighbor max cliques A_1, A_2, A_3 of B such that $\mathcal{M}_u = \{A_1, B, A_2\}$, $\mathcal{M}_v = \{A_2, B, A_3\}$ and $\mathcal{M}_w = \{A_3, B, A_1\}$. Let us assume B is of degree at least 4. Let C be a neighbor of B in T_G that is distinct from A_1, A_2 and A_3. According to Lemma 3.5 there must be a vertex $z \in V$ such that $B, C \in \mathcal{M}_z$ (for an illustration see Figure 8). By Lemma 3.11, we have $|\mathcal{M}_z| = 3$. W.l.o.g., let A_2 and A_3 be not contained in $\mathcal{M}_z$. Then $T_G[\mathcal{M}_v \setminus \mathcal{M}_z]$ is not connected, and we obtain a contradiction to Corollary 3.3. □

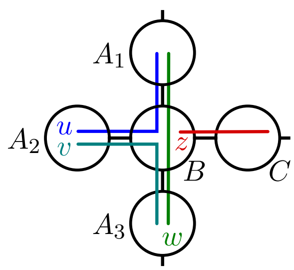


FIGURE 8. Illustration for the proof of Lemma 3.12

Corollary 3.13. *If a max clique $B \in \mathcal{M}$ has a fork triangle, then B is a fork clique.*

Proof. Let B be a max clique that has a fork triangle. Then the degree of B is 3 by Lemma 3.12. As B has a fork triangle, there exists a vertex $v \in B$ with $\mathcal{M}_v = \{B, A, A'\}$ for all neighbor max cliques A, A' of B with $A \neq A'$. Further, it follows from Lemma 3.11 that for every $v \in B$ there exist two neighbor max cliques A, A' of B with $A \neq A'$ such that $\mathcal{M}_v = \{B, A, A'\}$. □

Now we can prove Lemma 3.7 and show that each max clique of degree at least 3 in the clique tree T_G is a star clique or a fork clique.

Proof of Lemma 3.7. Let B be a max clique of degree at least 3. Suppose B is not a star clique. Then there exists a vertex $u \in B$ and two neighbor max cliques A_1, A_2 of B in T_G that also contain vertex u. Let C be a neighbor of B with $C \neq A_1$ and $C \neq A_2$. Since $\{B, C\}$ is an edge of T_G, there must be a vertex $w \in V$ such that $B, C \in \mathcal{M}_w$ according to Lemma 3.5 (see Figure 9 for an illustration). By Corollary 3.3, the graph $T_G[\mathcal{M}_u \setminus \mathcal{M}_w]$ must be connected. Thus, we have $A_1 \in \mathcal{M}_w$ or $A_2 \in \mathcal{M}_w$. Hence, $T_G[\mathcal{M}_u]$ and $T_G[\mathcal{M}_w]$

fork in B, and Lemma 3.10 implies that B has a fork triangle. It follows from Corollary 3.13 that B is a fork clique. □

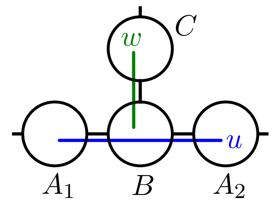


Figure 9. Illustration for the proof of Lemma 3.7

4. The Supplemented Clique Tree

In this section we define the supplemented clique tree of a connected chordal claw-free graph G. We obtain the supplemented clique tree by transferring the clique tree T_G into a directed tree and including some of the structural information about each max clique into the directed clique tree by means of an LO-coloring. We show that there exists a parameterized STC+C-transduction that defines for each connected chordal claw-free graph and every tuple of suitable parameters an isomorphic copy of the corresponding supplemented clique tree. In order to do this, we first present (parameterized) transductions for the clique tree and the directed clique tree. Throughout this section we let $\bar{x}, \bar{y}$ and $\bar{y}'$ be triples of structure variables.

4.1. Defining the Clique Tree in FO.

In a first step we present an FO-transduction $\Theta = (\theta_U(\bar{y}), \theta_{\approx}(\bar{y}, \bar{y}'), \theta_E(\bar{y}, \bar{y}'))$ that defines for each connected chordal claw-free graph G a tree isomorphic to the clique tree of G.

For now, let $G = (V, E)$ be a chordal claw-free graph, and let $\mathcal{M}$ be the set of max cliques of G. A triple $\bar{b} = (b_1, b_2, b_3) \in V^3$ *spans* a max clique $A \in \mathcal{M}$ if A is the only max clique that contains the vertices b_1, b_2 and b_3. Thus, $\bar{b}$ spans max clique $A \in \mathcal{M}$ if and only if $\mathcal{M}_{b_1} \cap \mathcal{M}_{b_2} \cap \mathcal{M}_{b_3} = \{A\}$. We call $\bar{b} \in V^3$ a *spanning triple* of G if $\bar{b}$ spans a max clique. We use spanning triples to represent max cliques. Note that this concept was already used in [Lau10] and [GGHL12] to represent max cliques of interval graphs.

Lemma 4.1. *Every max clique of a chordal claw-free graph is spanned by a triple of vertices.*

Proof. Let $T = (\mathcal{M}, \mathcal{E})$ be a clique tree of a chordal claw-free graph G. Let $B \in \mathcal{M}$ and let $v \in B$. By Lemma 3.1, the induced subgraph $T[\mathcal{M}_v]$ is a path $P = B_1, \ldots, B_l$. Suppose $B = B_i$. If $i > 1$, let u be a vertex in $B \setminus B_{i-1}$, and let w be a vertex in $B \setminus B_{i+1}$ if $i < l$. We let $u = v$ if $i = 1$, and we let $w = v$ if $i = l$. Then (u, v, w) spans max clique B: Clearly, $u, v, w \in B$. It remains to show, that there does not exist a max clique $A \in \mathcal{M}$ with $A \neq B$ and $u, v, w \in A$. Let us suppose such a max clique A exists. Since $v \in A$, max clique A is a vertex on path P. W.l.o.g., suppose $A = B_j$ for $j < i$. According to the clique intersection property, we have $u \in A \cap B \subseteq B_{i-1}$, a contradiction. □

As a direct consequence of Lemma 4.1, there exists an at most cubic number of max cliques in a chordal claw-free graph.

The following observations contain properties that help us to define the transduction Θ.

Observation 4.2. Let $G = (V, E)$ be a chordal claw-free graph. Let $\bar{v} = (v_1, v_2, v_3) \in V^3$. Then $\bar{v}$ is a spanning triple of G if, and only if, $\tilde{v}$ is a clique and $\{w_1, w_2\} \in E$ for all vertices $w_1, w_2 \in N(v_1) \cap N(v_2) \cap N(v_3)$ with $w_1 \neq w_2$.

Proof. Let $G = (V, E)$ be a chordal claw-free graph, and let $\bar{v} = (v_1, v_2, v_3) \in V^3$. First, suppose that $\bar{v}$ is a spanning triple. Then $\tilde{v}$ is a clique. Let us assume there exist vertices $w_1, w_2 \in N(v_1) \cap N(v_2) \cap N(v_3)$ with $w_1 \neq w_2$ such that there is no edge between w_1 and w_2. Then $\tilde{v} \cup \{w_1\}$ and $\tilde{v} \cup \{w_2\}$ are cliques but $\tilde{v} \cup \{w_1, w_2\}$ is not a clique. Thus, $\tilde{v} \cup \{w_1\}$ is a subset of a max clique C_1 with $w_2 \notin C_1$, and $\tilde{v} \cup \{w_2\}$ is a subset of a max clique C_2 with $w_1 \notin C_2$. Consequently, vertices v_1, v_2, v_3 are contained in more than one max clique, and therefore, $\bar{v}$ is no spanning triple, a contradiction.

Next, let us suppose that $\tilde{v}$ is a clique and that $\{w_1, w_2\} \in E$ for all vertices $w_1, w_2 \in N(v_1) \cap N(v_2) \cap N(v_3)$ with $w_1 \neq w_2$. Assume that v_1, v_2, v_3 are contained in two max cliques A and B. As A cannot be a subset of B, there exists a vertex $w_1 \in A \setminus B$. Now, $B \cup \{w_1\}$ cannot be a clique. Thus, there must exist a vertex $w_2 \in B$ that is not adjacent to w_1. Since w_1 is adjacent to all vertices in $A \setminus \{w_1\}$, we have $w_2 \in B \setminus A$. Consequently, w_1 and w_2 are vertices in $N(v_1) \cap N(v_2) \cap N(v_3)$ with $w_1 \neq w_2$ that are not adjacent, a contradiction. □

From the characterization of spanning triples in Observation 4.2, it follows that there exists an FO-formula $\theta_U(\bar{y})$ that is satisfied by a chordal claw-free graph $G = (V, E)$ and a triple $\bar{v} \in V^3$ if and only if $\bar{v}$ is a spanning triple of G.

Observation 4.3. Let $G = (V, E)$ be a chordal claw-free graph. Let A be a max clique of G, and let the triple $\bar{v} = (v_1, v_2, v_3) \in V^3$ span A. Then $w \in A$ if, and only if, $w \in \tilde{v}$ or $\{w, v_j\} \in E$ for all $j \in [3]$.

Proof. Let A be a max clique of a chordal claw-free graph $G = (V, E)$, and let $\bar{v} = (v_1, v_2, v_3) \in V^3$ span A. Clearly, if $w \in A$, then $w \in \tilde{v}$ or $\{w, v_j\} \in E$ for all $j \in [3]$. Further, $w \in A$ if $w \in \tilde{v}$. Thus, we only need to show that $w \in A$ if $\{w, v_j\} \in E$ for all $j \in [3]$. Suppose $\{w, v_j\} \in E$ for all $j \in [3]$. Then $\{v_1, v_2, v_3, w\}$ is a clique. Let B be a max clique with $\{v_1, v_2, v_3, w\} \subseteq B$. Since A is the only max clique that contains v_1, v_2, v_3, we have $B = A$. Hence, $w \in A$. □

Observation 4.3 yields that there further exists an FO-formula $\varphi_{\mathrm{mc}}(\bar{y}, z)$ that is satisfied for $\bar{v} \in V^3$ and $w \in V$ in a chordal claw-free graph $G = (V, E)$ if, and only if, $\bar{v}$ spans a max clique A and $w \in A$. We can use this formula to obtain an FO-formula $\theta_{\approx}(\bar{y}, \bar{y}')$ such that for all chordal claw-free graphs $G = (V, E)$ and all triples $\bar{v}, \bar{v}' \in V^3$ we have $G \models \theta_{\approx}(\bar{v}, \bar{v}')$ if, and only if, $\bar{v}$ and $\bar{v}'$ span the same max clique.

In the following we consider connected chordal claw-free graphs G. The next observation is a consequence of Lemma 3.5 and Lemma 3.2.

Observation 4.4. Let $G = (V, E)$ be a connected chordal claw-free graph, and $T_G = (\mathcal{M}, \mathcal{E})$ be the clique tree of G. Let $A, B \in \mathcal{M}$. Max cliques A and B are adjacent in T_G if, and only if, there exists a vertex $v \in V$ such that $v \in A \cap B$ and for all $C \in \mathcal{M}$ with $v \in C$ we have $C \nsubseteq A \cup B$.

Proof. Let $T_G = (\mathcal{M}, \mathcal{E})$ be the clique tree of a connected chordal claw-free graph $G = (V, E)$. Let $A, B \in \mathcal{M}$. By Lemma 3.5 there is an edge between two max cliques $A, B \in \mathcal{M}$ in T_G if, and only if, there exists a vertex $v \in V$ such that $A, B \in \mathcal{M}_v$ and there is an edge between A and B on the path $T[\mathcal{M}_v]$. Further, it follows from Lemma 3.2 that max cliques $A, B \in \mathcal{M}_v$ are adjacent precisely if there does not exist a max clique $C \in \mathcal{M}_v$ with $C \subseteq A \cup B$. □

It follows from Observation 4.4 that there exists an FO-formula $\theta_E(\bar{y},\bar{y}')$ that is satisfied for triples $\bar{v},\bar{v}' \in V^3$ in a connected chordal claw-free graph $G=(V,E)$ if, and only if, $\bar{v}$ and $\bar{v}'$ span adjacent max cliques.

It is not hard to see that $\Theta = (\theta_U, \theta_\approx, \theta_E)$ is an FO-transduction that defines for each connected chordal claw-free graph G a tree isomorphic to the clique tree of G.

Lemma 4.5. *There exists an* FO*-transduction* Θ *such that* $\Theta[G] \cong T_G$ *for all* $G \in \text{con-CCF}$.

4.2. **The Directed Clique Tree and its Definition in STC.** Now we transfer the clique tree into a directed tree and show that this directed clique tree can be defined in STC.

Let R be a leaf of the clique tree T_G. We transform T_G into a directed tree by rooting T_G at max clique R. We denote the resulting directed clique tree by $T_G^R = (\mathcal{M}, \mathcal{E}_R)$. Since R is a leaf of T_G, the following corollary is an immediate consequence of Lemma 3.7.

Corollary 4.6. *Let A be a max clique of a connected chordal claw-free graph G. If A is a vertex with at least two children in T_G^R, then A is a star clique or a fork clique.*

In the following we show that there exists a parameterized STC-transduction $\Theta'(\bar{x})$ which defines an isomorphic copy of T_G^R for each connected chordal claw-free graph G and triple $\bar{r} \in V^3$ that spans a leaf R of T_G.

Clearly, we can define an FO-formula $\theta'_{\text{dom}}(\bar{x})$ such that for all connected chordal claw-free graphs G and $\bar{r} \in V^3$ we have $G \models \theta'_{\text{dom}}(\bar{r})$ if, and only if, $\bar{r} \in V^3$ spans a leaf of T_G. Then θ'_{dom} defines the triples of parameters of transduction $\Theta'(\bar{x})$. Further, we let $\theta'_U(\bar{x},\bar{y}) := \theta_U(\bar{y})$ and $\theta'_\approx(\bar{x},\bar{y},\bar{y}') := \theta_\approx(\bar{y},\bar{y}')$. Finally, we let $\theta'_E(\bar{x},\bar{y},\bar{y}')$ be satisfied for triples $\bar{r},\bar{v},\bar{v}' \in V^3$ in a connected chordal claw-free graph $G=(V,E)$ if, and only if, $\bar{r}$, $\bar{v}$ and $\bar{v}'$ span max cliques R, A and A', respectively, and (A,A') is an edge in T_G^R. Note that (A,A') is an edge in T_G^R precisely if $\{A,A'\}$ is an edge in T_G and there exists a path between R and A in T_G after removing A'. Thus, formula θ'_E can be constructed in STC. We let $\Theta'(\bar{x}) := (\theta'_{\text{dom}}(\bar{x}), \theta'_U(\bar{x},\bar{y}), \theta'_\approx(\bar{x},\bar{y},\bar{y}'), \theta'_E(\bar{x},\bar{y},\bar{y}'))$, and conclude:

Lemma 4.7. *There exists a parameterized* STC*-transduction* $\Theta'(\bar{x})$ *such that* $\mathrm{Dom}(\Theta'(\bar{x}))$ *is the set of all pairs* $(G,\bar{r})$ *where* $G=(V,E) \in \text{con-CCF}$ *and* $\bar{r} \in V^3$ *spans a leaf R of T_G, and* $\Theta'[G,\bar{r}] \cong T_G^R$ *for all* $(G,\bar{r}) \in \mathrm{Dom}(\Theta'(\bar{x}))$ *where $\bar{r}$ spans the max clique R of G.*

4.3. **The Supplemented Clique Tree and its Definition in STC+C.** We now equip each max clique of the directed clique tree T_G^R with structural information. We do this by coloring the directed clique tree T_G^R with an LO-coloring. An LO-color is a binary relation on a linearly ordered set of basic color elements. Into each LO-color, we encode three numbers. Isomorphisms of LO-colored directed trees preserve the information that is encoded in the LO-colors. Thus, an LO-colored directed tree and its canon contain the same numbers encoded in their LO-colors. We call this LO-colored directed clique tree a supplemented clique tree. More precisely, let $G \in \text{con-CCF}$ and let R be a leaf of the clique tree T_G of G, then the *supplemented clique tree* S_G^R is the 5-tuple $(\mathcal{M}, \mathcal{E}_R, [0,|V|], \leq_{[0,|V|]}, L)$ where

- $(\mathcal{M}, \mathcal{E}_R)$ is the directed clique tree T_G^R of G,
- $\leq_{[0,|V|]}$ is the natural linear order on the set of basic color elements $[0,|V|]$,
- $L \subseteq \mathcal{M} \times [0,|V|]^2$ is the ternary color relation where
  - $(A,0,n) \in L$ iff n is the number of vertices in A that are not in any child of A in T_G^R,

– $(A, 1, n) \in L$ iff n is the number of vertices that are contained in A and in the parent of A in T_G^R if $A \neq R$, and $n = 0$ if $A = R$,
– $(A, 2, n) \in L$ iff n is the number of vertices in A that are in two children of A in T_G^R.[9]

In its structural representation the supplemented clique tree S_G^R corresponds to the 6-tuple $(\mathcal{M} \mathbin{\dot\cup} [0, |V|], \mathcal{M}, \mathcal{E}_R, [0, |V|], \leq_{[0,|V|]}, L)$.

Example 4.8. Figure 10 shows a supplemented clique tree, that is, a directed clique tree with its LO-coloring.

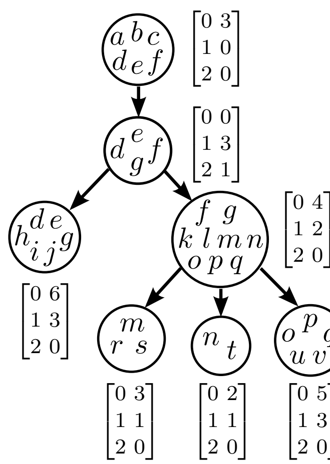


FIGURE 10. A supplemented clique tree S_G^R

The properties encoded in the colors of the max cliques are expressible in STC+C. Therefore, we can extend the parameterized STC-transduction $\Theta'(\bar{x})$ to a parameterized STC+C-transduction $\Theta''(\bar{x})$ that defines an LO-colored digraph isomorphic to S_G^R for every connected chordal claw-free graph G and triple $\bar{r} \in V^3$ that spans a leaf R of T_G.

Lemma 4.9. *There is a parameterized* STC+C*-transduction* $\Theta''(\bar{x})$ *such that* $\mathrm{Dom}(\Theta''(\bar{x}))$ *is the set of all pairs* $(G, \bar{r})$ *where* $G = (V, E) \in$ con-CCF *and* $\bar{r} \in V^3$ *spans a leaf* R *of* T_G*, and* $\Theta''[G, \bar{r}] \cong S_G^R$ *for all* $(G, \bar{r}) \in \mathrm{Dom}(\Theta''(\bar{x}))$ *where* $\bar{r}$ *spans the max clique* R *of* G.

Proof. We let $\Theta'(\bar{x}) := (\theta'_{\mathrm{dom}}(\bar{x}), \theta'_U(\bar{x}, \bar{y}), \theta'_{\approx}(\bar{x}, \bar{y}, \bar{y}'), \theta'_E(\bar{x}, \bar{y}, \bar{y}'))$ be the parameterized STC-transduction from Lemma 4.7. Then the domain $\mathrm{Dom}(\Theta'(\bar{x}))$ of $\Theta'(\bar{x})$ is the set of all pairs $(G, \bar{r})$ where $G = (V, E) \in$ con-CCF and $\bar{r} \in V^3$ spans a leaf R of T_G, and we have $\Theta'[G, \bar{r}] \cong T_G^R$ for all $(G, \bar{r}) \in \mathrm{Dom}(\Theta'(\bar{x}))$ where $\bar{r}$ spans the max clique R.

We can define a parameterized STC+C-counting transduction $\Theta^\#(\bar{x})$ as follows:
We let

$$\Theta^\#(\bar{x}) := \Big(\theta^\#_{\mathrm{dom}}(\bar{x}), \theta^\#_U(\bar{x}, \bar{y}), \theta^\#_{\approx}(\bar{x}, \bar{y}, \bar{y}'), \theta^\#_V(\bar{x}, \bar{y}), \theta^\#_E(\bar{x}, \bar{y}, \bar{y}'),$$
$$\theta^\#_M(\bar{x}, p), \theta^\#_{\trianglelefteq}(\bar{x}, p, p'), \theta^\#_L(\bar{x}, \bar{u}, p, p'), \Big),$$

[9] Let A be a max clique and n be the number of vertices in A that are in two children of A in T_G^R. Notice that according to Corollary 4.6, A is a fork clique if and only if $n > 0$.

where

$$\theta^{\#}_{\mathrm{dom}}(\bar{x}) := \theta'_{\mathrm{dom}}(\bar{x}) \qquad \theta^{\#}_{V}(\bar{x},\bar{y}) := \theta'_{U}(\bar{x},\bar{y}) \qquad \theta^{\#}_{M}(\bar{x},p) := \top$$
$$\theta^{\#}_{U}(\bar{x},\bar{y}) := \theta'_{U}(\bar{x},\bar{y}) \qquad \theta^{\#}_{E}(\bar{x},\bar{y},\bar{y}') := \theta'_{E}(\bar{x},\bar{y},\bar{y}') \qquad \theta^{\#}_{\trianglelefteq}(\bar{x},p,p') := p \leq p'$$
$$\theta^{\#}_{\approx}(\bar{x},\bar{y},\bar{y}') := \theta'_{\approx}(\bar{x},\bar{y},\bar{y}')$$

and

$$\theta^{\#}_{L}(\bar{x},\bar{y},p,p') := \varphi_0(\bar{x},\bar{y},p,p') \vee \varphi_1(\bar{x},\bar{y},p,p') \vee \varphi_2(\bar{x},\bar{y},p,p').$$

We let $\varphi_0(\bar{x},\bar{y},p,p')$, $\varphi_1(\bar{x},\bar{y},p,p')$ and $\varphi_2(\bar{x},\bar{y},p,p')$ be STC+C-formulas such that for all $G = (V,E) \in$ con-CCF, all triples $\bar{r} \in V^3$ that span a leaf R of T_G, all $\bar{v} \in V^3$ and all $m,n \in N(G)$:

- $G \models \varphi_0[\bar{r},\bar{v},m,n]$ iff $m = 0$, the triple $\bar{v}$ spans a max clique A of G, and n is the number of vertices in A that are not in any child of A in T^R_G.
- $G \models \varphi_1[\bar{r},\bar{v},m,n]$ iff $m = 1$, the triple $\bar{v}$ spans a max clique A of G, and n is the number of vertices that are contained in A and in the parent of A in T^R_G if $A \neq R$, and $n = 0$ if $A = R$.
- $G \models \varphi_2[\bar{r},\bar{v},m,n]$ iff $m = 2$, the triple $\bar{v}$ spans a max clique A of G, and n is the number of vertices in A that are in at least two children of A in T^R_G.

Let $\varphi_{\mathrm{mc}}(\bar{y},z)$ be the FO-formula from Section 4, which is satisfied for $\bar{v} \in V^3$ and $w \in V$ in a chordal claw-free graph $G = (V,E)$ if, and only if, $\bar{v}$ spans a max clique A and $w \in A$. Then $\varphi_0(\bar{x},\bar{y},p,p')$, for example, can be defined as follows:

$$\begin{aligned}\varphi_0(\bar{x},\bar{y},p,p') := \;& \forall q\; p \leq q \;\wedge\; \theta'_U(\bar{x},\bar{y}) \;\wedge \\ & \#z\Big(\varphi_{\mathrm{mc}}(\bar{y},z) \wedge \forall \bar{y}'\, \big(\theta'_E(\bar{x},\bar{y},\bar{y}') \rightarrow \neg\varphi_{\mathrm{mc}}(\bar{y}',z)\big)\Big) = p'\,.\end{aligned}$$

It should be clear how to define $\varphi_1(\bar{x},\bar{y},p,p')$ and $\varphi_2(\bar{x},\bar{y},p,p')$.

It is not hard to see that $\Theta^{\#}(\bar{x})$ is a parameterized STC+C-counting transduction whose domain $\mathrm{Dom}(\Theta^{\#}(\bar{x}))$ is the set of all pairs $(G,\bar{r})$ where $G = (V,E) \in$ con-CCF and $\bar{r} \in V^3$ spans a leaf R of T_G, and which satisfies $\Theta^{\#}[G,\bar{r}] \cong S^R_G$ for all $(G,\bar{r}) \in \mathrm{Dom}(\Theta^{\#}(\bar{x}))$ where $\bar{r}$ spans the max clique R of G. Now Lemma 4.9 follows directly from Proposition 2.5. □

5. Canonization

In this section we prove that there exists a parameterized LREC$_=$-canonization of the class of connected chordal claw-free graphs, which is the main result of this paper.

Theorem 5.1. *The class of chordal claw-free graphs admits* LREC$_=$*-definable canonization.*

Proof of Theorem 5.1. We prove that there exists a parameterized LREC$_=$-canonization of the class of connected chordal claw-free graphs. Then Proposition 2.6 implies that there also exists one for the class of chordal claw-free graphs.

Thus, let us show that there exists a parameterized LREC$_=$-canonization of con-CCF. By Lemma 4.9 there exists a parameterized STC+C-, and therefore, LREC$_=$-transduction $\Theta''(\bar{x})$ such that $\Theta''[G,\bar{r}]$ is isomorphic to the LO-colored directed tree S^R_G for all connected chordal claw-free graphs $G = (V,E)$ and all triples $\bar{r} \in V^3$ that span a leaf R of T_G. Further, there exists an LREC$_=$-canonization Θ^{LO} of the class of LO-colored directed trees according to Theorem 2.7. We show that there also exists an LREC$_=$-transduction Θ^K which defines for each canon $K(S_G^R)$ of a supplemented clique tree S_G^R of $G \in$ con-CCF the canon $K(G)$ of G. Then we can compose the (parameterized) $\mathsf{LREC}_=$-transductions $\Theta''(\bar{x})$, Θ^{LO} and Θ^K (see Figure 11) to obtain a parameterized $\mathsf{LREC}_=$-canonization of the class of connected chordal claw-free graphs (Proposition 2.3).

$$\begin{tikzcd}
\boxed{\begin{array}{c} G \in \text{con-CCF}, \bar{r} \in V^3 \\ (\bar{r} \text{ spans a leaf } R \text{ of } T_G) \end{array}} \arrow[r, "\Theta''(\bar{x})"] & \boxed{\Theta''[G,\bar{r}]} \arrow[r, "\Theta^{\mathrm{LO}}"] \arrow[d, phantom, "\cong"] & \boxed{\Theta^{\mathrm{LO}}[\Theta''[G,\bar{r}]]} \arrow[r, "\Theta^K"] \arrow[d, phantom, "="] & \boxed{\Theta^K[\Theta^{\mathrm{LO}}[\Theta''[G,\bar{r}]]]} \arrow[d, phantom, "="] \\
& \boxed{S_G^R} & \boxed{K(S_G^R)} & \boxed{K(G)}
\end{tikzcd}$$

FIGURE 11. Overview of the composition of (parameterized) transductions

We let $\mathsf{LREC}_=[\{V, E, M, \trianglelefteq, L, \leq\}, \{E, \leq\}]$-transduction $\Theta^K = (\theta_V(p), \theta_E(p,p'), \theta_{\leq}(p,p'))$ define for each canon $K(S_G^R) = (U_K, V_K, E_K, M_K, \trianglelefteq_K, L_K, \leq_K)$ of a supplemented clique tree of $G \in$ con-CCF an ordered copy $K(G) = (V'_K, E'_K, \leq'_K)$ of $G = (V, E)$. We let V'_K be the set $[|V|]$, and $\leq'_K$ be the natural linear order on $[|V|]$. As the set of basic color elements of S_G^R is $[0, |V|]$, the set M_K of basic color elements of the canon $K(S_G^R)$ contains exactly $|V|+1$ elements. Hence, we can easily define the vertex set of $K(G)$ by counting the number of basic color elements of $K(S_G^R)$. We let $\varphi_V(p) := \exists q \big(p \leq q \land p \neq 0 \land \#x\, M(x) = q \big)$. Further, we let $\theta_{\leq}(p,p') := p \leq p'$. In order to show that there exists an $\mathsf{LREC}_=$-formula $\theta_E(p,p')$, which defines the edge relation of $K(G)$, we exploit the property that $\mathsf{LREC}_=$ captures LOGSPACE on ordered structures (Corollary 2.9), and show that there exists a logarithmic-space algorithm that computes the edge relation of $K(G)$, instead. According to Lemma 5.2 there exists a logarithmic-space algorithm that computes the set of max cliques of $K(G)$. As every edge is a subset of some max clique and every two distinct vertices in a max clique are adjacent, such a logarithmic-space algorithm can easily be extended to a logarithmic-space algorithm that decides whether a pair of numbers is an edge of $K(G)$. $\square$

Lemma 5.2. *There exists a logarithmic-space algorithm that, given the canon $K(S_G^R)$ of a supplemented clique tree of a connected chordal claw-free graph G, computes the set of max cliques of an ordered copy $K(G)$ of G.*

In the following we briefly sketch the algorithm. A detailed proof of Lemma 5.2 follows afterwards.

The algorithm performs a post-order tree traversal on the underlying tree of the canon $K(S_G^R) = (U_K, V_K, E_K, M_K, \trianglelefteq_K, L_K, \leq_K)$ of the supplemented clique tree S_G^R. Let $m_1, \dots, m_{|\mathcal{M}|}$ be the post-order traversal sequence. Each vertex $m_k \in V_K$ of the canon $K(S_G^R)$ corresponds to a vertex, i.e., a max clique $A_k \in \mathcal{M}$, in the supplemented clique tree S_G^R. We call $A_1, \dots, A_{m_{|\mathcal{M}|}}$ a *transferred traversal sequence*. For all $k \in \{1, \dots, |\mathcal{M}|\}$, starting with $k = 1$, the algorithm constructs for $m_k \in V_K$ a copy $B_{m_k} \subseteq [|V|]$ of A_k.

From the information encoded in the colors of the vertices of $K(S_G^R)$, we know the number of vertices in A_k that are not in any max clique that occurs before A_k in the transferred traversal sequence. For these vertices, we add the smallest numbers of $[|V|]$ to B_{m_k} that were not used before. We also use the information in the colors to find out how many vertices of A_k are in a max clique A_i that occurs before A_k in the transferred traversal sequence, and to determine what numbers these vertices were assigned to. These numbers are added to B_{m_k} as well.

We will see that the algorithm computes the max cliques $B_{m_1}, \dots, B_{m_{k-1}}$ in logarithmic space.

In the remainder of this section we prove Lemma 5.2. We start with looking at the structure of the required algorithm, and focus on its basic idea. Then we make necessary observations, and finally present the algorithm. Afterwards, we prove its correctness and show that it only needs logarithmic space.

In the following let $G = (V, E)$ be a connected chordal claw-free graph and S_G^R be a supplemented clique tree of G. Further, let $K(S_G^R) = (U_K, V_K, E_K, M_K, \trianglelefteq_K, L_K, \leq_K)$ be the canon of S_G^R. Without loss of generality, we assume that the set of basic color elements M_K is $[0, |V|]$ and that $\trianglelefteq_K$ is the natural linear order $\leq_{[0,|V|]}$ on $[0, |V|]$.

The goal is to define the max cliques of an ordered copy of G. We denote this ordered copy by $K(G) = (V_K', E_K', \leq_K')$, and let V_K' be the set $[|V|]$ and $\leq_K'$ be the natural linear order on $[|V|]$.

Post-Order Depth-First Tree Traversal. The algorithm uses post-order traversal (see, e.g., [Sed02]) on the underlying directed tree of $K(S_G^R)$ to construct the max cliques of the canon $K(G)$ of G. Like pre-order and in-order traversal, post-order traversal is a type of depth-first tree traversal, that specifies a linear order on the vertices of a tree.

Note that the universe of the canon of the supplemented clique tree is linearly ordered. Thus, we have a linear order on the children of a vertex, and we assume the children of a vertex to be given in that order.

In the following we summarize the logarithmic-space algorithm for *depth-first traversal* described by Lindell in [Lin92]. We start at the root. For every vertex of the tree we have three possible moves:

- **down**: go down to the first child, if it exists
- **over**: move over to the next sibling, if it exists
- **up**: buck up to the parent, if it exists

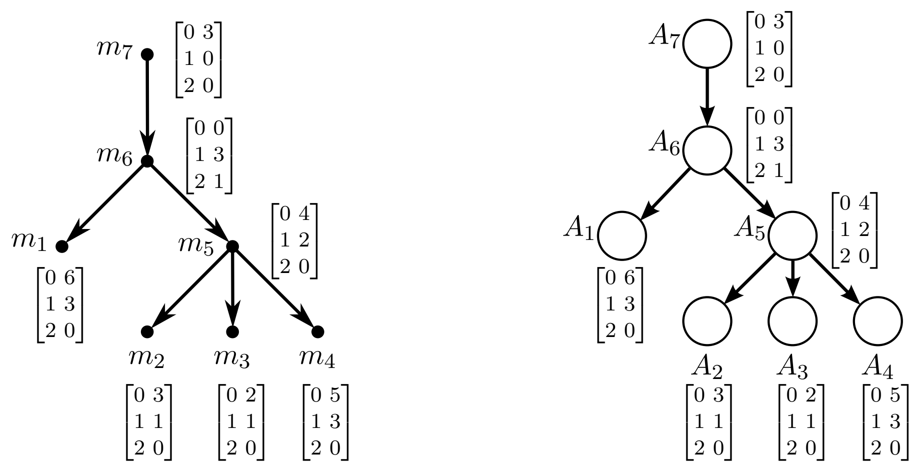
If our last move was **down**, **over** or there was no last move, which means we are visiting a new vertex, then we perform the first move out of **down**, **over** or **up** that succeeds. If our last move was **up**, then we are backtracking, and we call **over** if it is possible or else **up**. Note that at each step we only need to remember our last move and the current vertex. Therefore, we only need logarithmic space for depth-first traversal.

The *post-order traversal sequence* consists of every vertex we visit during the depth-first traversal in order of its last visit. It follows that we obtain the post-order traversal sequence by successively adding all vertices visited during depth-first traversal that are not followed by the move **down**. Thus, we can perform post-order traversal in logarithmic space.[10]

Let $m_1, \dots, m_{|\mathcal{M}|}$ be the post-order traversal sequence of the underlying directed tree of the canon $K(S_G^R)$. We know that there exists an isomorphism I between $K(S_G^R)$ and S_G^R. For all $k \in [|\mathcal{M}|]$ the isomorphism I maps the vertex m_k of $K(S_G^R)$ to a vertex, i.e., a max clique $A_k := I(m_k)$, of the supplemented clique tree S_G^R. Notice that the vertices m_k and A_k have the same color. We call $A_1, \dots, A_{|\mathcal{M}|}$ the traversal sequence *transferred*

[10] We also obtain the *post-order traversal* of an ordered directed tree T with root r recursively as follows: Let d be the out-degree of r. For all $i \in \{1, \dots, d\}$, in increasing order, perform a post-order traversal on the subtree rooted at child i of the root. Afterward, visit r. Note that this does not correspond to a logarithmic-space algorithm.

by isomorphism I. The isomorphism I also transfers the ordering of the children of a vertex. A sequence $A_1, \dots, A_{|\mathcal{M}|}$ is a *transferred (post-order) traversal sequence* if there exists an isomorphism I between $K(S_G^R)$ and S_G^R, and $A_1, \dots, A_{|\mathcal{M}|}$ is the traversal sequence transferred by isomorphism I. Figure 12 shows an example of a canon $K(S_G^R)$ and its post-order traversal sequence $m_1, \dots, m_{|\mathcal{M}|}$,[11] and the corresponding supplemented clique tree S_G^R and its transferred traversal sequence $A_1, \dots, A_{|\mathcal{M}|}$.



(A) Canon $K(S_G^R)$ and its post-order traversal sequence $m_1, \dots, m_7$

(B) The supplemented clique tree S_G^R and a transferred traversal sequence $A_1, \dots, A_7$

FIGURE 12

Clearly, in the post-order traversal sequence of a tree, a proper descendant of a vertex v occurs before the vertex v. Regarding the supplemented clique tree S_G^R, this means:

Observation 5.3. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred post-order traversal sequence on S_G^R, and let $i, i' \in [|\mathcal{M}|]$. If max clique A_i is a proper descendant of max clique $A_{i'}$ in T_G^R, then $i < i'$.

Intersections of Max Cliques with Preceding Max Cliques in Transferred Post-Order Traversal Sequences. We traverse the underlying directed tree of $K(S_G^R)$ in post-order, and we construct the max cliques of the canon $K(G)$ of G during this post-order traversal. So for each vertex m_k of the directed tree we construct a clique $B_{m_k} \subseteq [|V|]$. The clique B_{m_k} will be the max clique of $K(G)$ that corresponds to max clique A_k of graph G.

In order to construct these cliques B_{m_k} during the traversal of the underlying directed tree of $K(S_G^R)$, we have to decide on numbers for all vertices that are supposed to be in such a clique. The numbering happens according to the post-order traversal sequence. The hard part will be to detect which vertices have already occurred in a clique corresponding to a vertex m_i we have visited before reaching m_k, and to determine the numbers they were assigned to. Then we can choose new numbers for newly occurring vertices and reuse the

[11] In fact, Figure 12A shows the canon of the supplemented clique tree depicted in Figure 10.

numbers that correspond to vertices that have occurred before. Thus, in the following we take a transferred post-order traversal sequence $A_1, \dots, A_{|\mathcal{M}|}$ and study the intersection of a max clique A_k with max cliques that precede A_k in the transferred traversal sequence.

An important observation in this respect is that if A_k is a fork clique, then the vertices in A_k only occur in the two children and the parent max clique of fork clique A_k (Corollary 3.13). Thus, apart from the two children of A_k the vertices in A_k are not contained in any other max clique previously visited in the transferred traversal sequence. Further, each vertex in A_k occurs in at least one child max clique of A_k. Hence, each vertex in A_k is contained in a max clique that was visited before.

If max clique A_k is not a fork clique, then it has only one child or is a star clique (Lemma 3.7). Thus, the vertices in A_k occur in no more than one child max clique of A_k. Observation 5.5 shows that each vertex $v \in A_k$ that occurs in a max clique that is visited before non-fork clique A_k in the transferred traversal sequence is either contained in exactly one child of A_k or in the first child of a fork clique A_l if A_k is the second child of A_l.

Observation 5.4. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred post-order traversal sequence of S_G^R. Let $k \in [|\mathcal{M}|]$ and let $v \in A_k$. If there exists a $j < k$ such that $v \in A_j$ and A_j is not a descendant of A_k in the underlying directed tree T_G^R of S_G^R, then A_j is the first and A_k the second child of a fork clique.

Proof. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred post-order traversal sequence of S_G^R. Let $j, k \in [\mathcal{M}]$ with $j < k$ and let $v \in A_j \cap A_k$. Further, suppose that A_j is not a descendant of A_k. As $j < k$, max clique A_k also cannot be a proper descendant of A_j by Observation 5.3. Consequently, the smallest common ancestor A_l of A_j and A_k must be a proper ancestor of A_j and A_k. Clearly, A_l has at least two children. Corollary 4.6 yields that A_l is either a star or a fork clique. According to the clique intersection property vertex v is contained in A_l and every max clique on the path between A_j and A_k. Thus, A_l must be a fork clique, and $T_G[\mathcal{M}_v]$ is a path of length 3. Therefore, A_j and A_k are the children of fork clique A_l. Since $j < k$, max clique A_j is the first and A_k the second child of A_l. □

Observation 5.5. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred post-order traversal sequence of S_G^R. Let $k \in [|\mathcal{M}|]$. Suppose that A_k is not a fork clique, and let $v \in A_k$. If there exists a $j < k$ such that $v \in A_j$, then there exists exactly one $i \in [|\mathcal{M}|]$ such that $v \in A_i$ and

(1) A_i is a child of A_k or
(2) A_i is the first child of a fork clique and A_k the second one.

Proof. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred post-order traversal sequence of S_G^R. Let $j, k \in [\mathcal{M}]$ with $j < k$ and let $v \in A_j \cap A_k$. Suppose that A_k is not a fork clique. If A_j is a descendant of A_k, then there exists an $i \in [|\mathcal{M}|]$ such that $v \in A_i$ and A_i is a child of A_k by the clique intersection property. If A_j is not a descendant of A_k, then by Observation 5.4 there exists an $i \in [|\mathcal{M}|]$, that is, $i = j$, such that A_i is the first and A_k the second child of a fork clique. Thus, there exists an $i \in [|\mathcal{M}|]$ such that $v \in A_i$ and property 1 or 2 is satisfied. Now, let us assume there exist $i_1, i_2 \in [|\mathcal{M}|]$ with $i_1 \neq i_2$ such that for all $m \in [2]$ we have $v \in A_{i_m}$ and

(1) A_{i_m} is a child of A_k or
(2) A_{i_m} is the first child of a fork clique and A_k the second one.

Clearly, A_{i_1} and A_{i_2} cannot be both the first child of a fork clique.

Now, let us consider the case, where A_{i_1} and A_{i_2} are children of A_k. Since A_k has at least two children and is not a fork clique, it must be a star clique by Corollary 4.6. However, v is contained in A_{i_1}, A_k and A_{i_2}. Therefore, A_k cannot be a star clique, a contradiction.

It remains to consider the case where, w.l.o.g., A_{i_1} is a child of A_k, and A_{i_2} is the first child of a fork clique A_l and A_k the second one. As $v \in A_{i_2}$ and $v \in A_{i_1}$, the clique intersection property implies that $v \in A_k$ and $v \in A_l$. Since $v \in A_l$ and $|\mathcal{M}_v| > 3$, we obtain a contradiction to A_l being a fork clique. $\square$

Now, let us summarize what we know about the intersection of a max clique with preceding max cliques in a transferred traversal sequence. If A_k is a fork clique, then we know the vertices of A_k all occur in its two children, which occur before A_k within a transferred traversal sequence. If A_k is not a fork clique, then by Observation 5.5 the vertices in A_k that occur in max cliques before A_k within a transferred traversal sequence are precisely the vertices in the pairwise intersection of A_k with its children, and the intersection of A_k with its sibling if A_k is the second child of a fork clique. Further, Observation 5.5 yields that these intersections are disjoint sets of vertices.

Algorithm to Construct the Cliques B_{m_j}. We now include the new knowledge about the intersection of max cliques with preceding max cliques in a transferred traversal sequence into our construction of the sets B_{m_j}. For the numbers in each clique B_{m_j} where m_j does not corresponds to the second child of a fork clique, we maintain the property that if a number $l \in B_{m_j}$ is contained in more ancestors of B_{m_j} than a number $l' \in B_{m_j}$, then $l > l'$. Thus, if B_{m_j} is a child of a clique $B_{m_{j'}}$, then the intersection $B_{m_j} \cap B_{m_{j'}}$ contains precisely the $|B_{m_j} \cap B_{m_{j'}}|$ largest numbers of B_{m_j}. In the following we present an algorithm that computes the sets B_{m_j}.

During the algorithm, we need to remember or compute a couple of values: At each step of our traversal, we let `count` be the total number of vertices we have created so far. We update this number *after* visiting a vertex m_k in the post-order traversal sequence $m_1, \dots, m_{|\mathcal{M}|}$ of the underlying directed tree of $K(S_G^R)$. Sometimes we need to recompute the number of vertices created up until after the visit of a vertex m_i with $i < k$. We let `count`(m_i) denote this number. Further, we exploit the information contained in the color of a vertex m. We let

- `in0children`(m) be the number of vertices that are contained in the max clique represented by m and are not contained in any max cliques corresponding to children of m,
- `inparent`(m) be the number of vertices that are contained in the max clique represented by m and the max clique represented by the parent of m (if m is the root of the tree, then `inparent`(m) will be 0), and
- `in2children`(m) be the number of vertices that are contained in the max clique corresponding to m and in at least two max cliques represented by children of m.

We also need the following boolean values. Note that they can be easily obtained from the color of a vertex, as well.

- `isforkclique`(m) which indicates whether m corresponds to a fork clique, and
- `isforkchild2`(m) which indicates whether m is the second child of a vertex corresponding to a fork clique.

With help of the above values, we can complete the algorithm. Thus, let us describe the algorithm at a vertex m during the post-order traversal. The algorithm distinguishes between the following cases. For each case we list the numbers belonging to clique B_m, and indicate the values used to determine the numbers in B_m.

1. Node m corresponds to a fork clique ($\texttt{isforkclique}(m) = \text{true}$).
Let m' be the first child of node m, and m'' be the second one. We determine $\texttt{count}(m')$, and since $\texttt{count}(m'') = \texttt{count}$, we already know $\texttt{count}(m'')$. Further, we need $\texttt{inparent}(m')$ and $\texttt{inparent}(m'')$, and $\texttt{in2children}(m)$. We let B_m be the set of numbers in

$$[\texttt{count}(m') - \texttt{inparent}(m') + 1,\ \texttt{count}(m')] \quad \text{and}$$
$$[\texttt{count}(m'') - \texttt{inparent}(m'') + \texttt{in2children}(m) + 1,\ \texttt{count}(m'')].$$

We do not increase $\texttt{count}$.

2. Node m does not correspond to a fork clique ($\texttt{isforkclique}(m) = \text{false}$).
Let $m_1, \dots, m_k$ be the children of m where $k \geq 0$. Now for all $j \in [k]$ we determine $\texttt{isforkclique}(m_j)$, and distinguish between the following two cases.

(a) $\texttt{isforkclique}(m_j) = \text{false}$:
We determine $\texttt{count}(m_j)$ and $\texttt{inparent}(m_j)$ and we add to B_m the numbers in

$$[\texttt{count}(m_j) - \texttt{inparent}(m_j) + 1,\ \texttt{count}(m_j)]$$

(b) $\texttt{isforkclique}(m_j) = \text{true}$:
Let m'_j and m''_j be the children of m_j. We add to B_m the numbers in

$$[\texttt{count}(m'_j) - \texttt{inparent}(m'_j) + \texttt{in2children}(m_j) + 1,\ \texttt{count}(m'_j)] \quad \text{and}$$
$$[\texttt{count}(m''_j) - \texttt{inparent}(m''_j) + \texttt{in2children}(m_j) + 1,\ \texttt{count}(m''_j)].$$

Further, we determine $\texttt{isforkchild2}(m)$ and depending on the value of it, we do the following.

(c) $\texttt{isforkchild2}(m) = \text{false}$:
We increase $\texttt{count}$ by $\texttt{in0children}(m)$, and add to B_m the vertices in

$$[\texttt{count} - \texttt{in0children}(m) + 1,\ \texttt{count}].$$

(d) $\texttt{isforkchild2}(m) = \text{true}$:
Let p be the parent of m, and let m' be the first sibling of m. We increase $\texttt{count}$ by $\texttt{in0children}(m) - \texttt{in2children}(p)$. We add to B_m the vertices in the intervals

$$[\texttt{count}(m') - \texttt{inparent}(m') + 1, \texttt{count}(m') - \texttt{inparent}(m') + \texttt{in2children}(p)],$$
$$[\texttt{count} - \texttt{in0children}(m) + \texttt{in2children}(p) + 1,\ \texttt{count}].$$

In the following we illustrate the algorithm with an example.

Example 5.6. The algorithm can be applied to the canon $K(S^R_G)$ depicted in Figure 12a. Figure 13 shows the computed values at each step of the algorithm. It also shows the cliques B_{m_i} for all i.

i	m_i	Case	B_{m_i}	`count`
				0
1	m_1	2(c)	$[1, 6]$	6
2	m_2	2(c)	$[7, 9]$	9
3	m_3	2(c)	$[10, 11]$	11
4	m_4	2(c)	$[12, 16]$	16
5	m_5	2(a) for m_2 2(a) for m_3 2(a) for m_4 2(d)	$[9, 9]$ $[11, 11]$ $[14, 16]$ $[4, 4] \cup [17, 19]$	19
6	m_6	1	$[4, 6] \cup [19, 19]$	19
7	m_7	2(b) for m_6 2(c)	$[5, 6] \cup [19, 19]$ $[20, 22]$	22

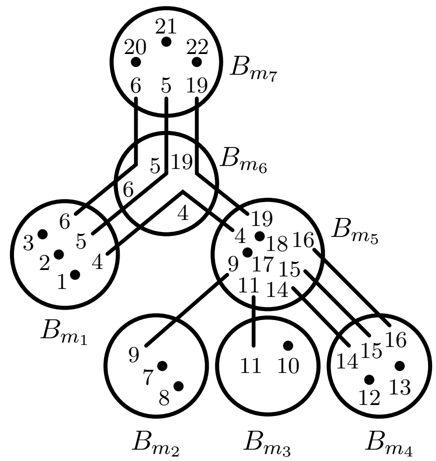


FIGURE 13. Application of the algorithm to the example in Figure 12A

Correctness of the Algorithm. We show that the presented algorithm returns the max cliques of an ordered copy of G. In order to do this, we prove that there exists a bijection h between V and $[|V|]$, so that for all $k \in [|\mathcal{M}|]$ we have $h(A_k) = B_{m_k}$. Then h is a graph isomorphism between G and the graph (V'_K, E'_K) where $V'_K = [|V|]$ and $\{v, v'\} \in E'_K$ iff $v \neq v'$ and there exists a $k \in [|\mathcal{M}|]$ such that $v, v' \in B_{m_k}$. Thus, $K(G) = (V'_K, E'_K, \leq'_K)$, where $\leq'_K$ is the natural linear order on $[|V|]$, is an ordered copy of G.

We show the existence of bijection h with help of the lemma below. The lemma is proved by induction along the post-order traversal sequence. First, we introduce definitions that are used in the lemma.

Let T^R_G be the underlying directed clique tree of the supplemented clique tree S^R_G. For all max cliques $A \in \mathcal{M}$ and for all $v \in A$ we let $\#\mathrm{anc}_A(v)$ be the number of max cliques in T^R_G that contain vertex v and are an ancestor of A. Clearly, for every vertex $v \in A$ the number $\#\mathrm{anc}_A(v)$ is at least 1. Let $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred traversal sequence. For $i \in [|\mathcal{M}|]$ and $c \in [2]$ let S^c_i be the set of vertices v of max clique A_i, where $\#\mathrm{anc}_{A_i}(v) > c$. Thus, if max clique A_i has a parent max clique P_i in T^R_G, then S^1_i is the set of vertices in $A_i \cap P_i$. Hence, $\texttt{inparent}(m_i) = |S^1_i|$. If again P_i has a parent in T^R_G, then S^2_i is the subset of vertices of A_i which are contained in P_i and the parent of P_i. For example, if A_l is a fork clique with children A_i and A_j, then A_l is the disjoint union of S^1_i and S^2_j. Further, if $A_{l'}$ is the parent max clique of fork clique A_l, then $A_{l'}$ is the disjoint union of S^2_i and S^2_j.

Lemma 5.7. *Let $m_1, \dots, m_{|\mathcal{M}|}$ be the post-order traversal sequence of the underlying directed tree of $K(S^R_G)$. Further, let $B_{m_1}, \dots, B_{m_{|\mathcal{M}|}}$ be the cliques computed by the algorithm and $A_1, \dots, A_{|\mathcal{M}|}$ be a transferred traversal sequence. Then, for all $l \in [|\mathcal{M}|]$ there exists a bijection h_l between $A_1 \cup \dots \cup A_l$ and $[\texttt{count}(m_l)]$, such that for all $i \in [l]$ we have*

(1) *$h_l(A_i) = B_{m_i}$,*
(2) *$\#\mathrm{anc}_{A_i}(v) \leq \#\mathrm{anc}_{A_i}(v')$ for all vertices $v, v' \in A_i$ with $h_l(v) \leq h_l(v')$ if A_i is neither a fork clique nor the second child of a fork clique,*

(3) $h_l(S^1_i) = [\texttt{count}(m_i) - \texttt{inparent}(m_i) + 1,\ \texttt{count}(m_i)]$ *if A_i is neither a fork clique nor the second child of a fork clique, and*

(4) $h_l(S^2_i) = [\texttt{count}(m_i) - \texttt{inparent}(m_i) + \texttt{in2children}(p_i) + 1,\ \texttt{count}(m_i)]$ *if A_i is the second child of a fork clique, where p_i is the parent of m_i.*

Proof. Let $m_1, \ldots, m_{|\mathcal{M}|}$ be the post-order traversal sequence, $B_{m_1}, \ldots, B_{m_{|\mathcal{M}|}}$ be the cliques computed by the algorithm and $A_1, \ldots, A_{|\mathcal{M}|}$ be a transferred traversal sequence. We prove Lemma 5.7 by induction on $l \in [0, |\mathcal{M}|]$. Notice that $l = 0$ is not included in the lemma, but we extend it to $l = 0$. Although there does not actually exist a vertex m_0, we let $\texttt{count}(m_0)$ be 0. This makes sense, since 0 is the initial value of $\texttt{count}$. We let $h_0 \colon \emptyset \to \emptyset$ be the empty mapping. Clearly, h_0 meets all the requirements. Now suppose $l > 0$ and let there be a bijection h_{l-1} with properties 1 to 4 for all $i \in [l-1]$. We show the existence of bijection h_l.

First, let us consider the case where m_l corresponds to a fork clique. Clearly, A_l is a subset of the set of vertices occurring in A_l's children, and $\texttt{count}(m_l) = \texttt{count}(m_{l-1})$. Thus, we let $h_l := h_{l-1}$, and we know by inductive assumption that it is a bijection. By inductive assumption we also know that h_l satisfies properties 1 to 4 for all $i < l$. Therefore, it remains to show these properties for $i = l$. As A_l is a fork clique, and cannot be the second child of a fork clique, properties 2, 3 and 4 trivially hold for $i = l$. Thus, we only have to show that h_l satisfies property 1 for $i = l$, that is, that $h_l(A_l) = B_{m_l}$.

So let us prove that $h_l(A_l) = B_{m_l}$. Let m_i and m_j with $i < j < l$ be respectively the first and the second child of m_l. Since m_i cannot correspond to a fork clique or to the second child of a fork clique, we have

$$h_l(S^1_i) = [\texttt{count}(m_i) - \texttt{inparent}(m_i) + 1,\ \texttt{count}(m_i)]$$

by inductive assumption. Analogously, we know

$$h_l(S^2_j) = [\texttt{count}(m_j) - \texttt{inparent}(m_j) + \texttt{in2children}(m_l) + 1,\ \texttt{count}(m_j)]$$

because the vertex m_j corresponds to the second child of a fork clique. We obtain that $B_{m_l} = h_l(S^1_i) \cup h_l(S^2_j)$. As A_l is a fork clique, A_l is the disjoint union of S^1_i and S^2_j. Hence, we have $B_{m_l} = h_l(A_l)$.

Next, suppose m_l is a vertex that does not correspond to a fork clique. By Observation 5.5 we know that there are $\texttt{in0children}(m_l)$ vertices in $A'_l := A_l \setminus \bigcup_{i<l} A_i$ if A_l is not the second child of a fork clique, and $\texttt{in0children}(m_l) - \texttt{in2children}(m_{l+1})$ vertices in A'_l if A_l is the second child of a fork clique (then m_{l+1} is the parent of m_l). Thus, A'_l and the set B'_{m_l} of newly occurring numbers in B_{m_l} have the same cardinality. We let h_l be an extension of h_{l-1} that bijectively maps the vertices in A'_l to the numbers in B'_{m_l} such that $h_l(v) \leq h_l(v')$ implies $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$ for all $v, v' \in A'_l$. Then h_l is a bijection between $A_1 \cup \cdots \cup A_l$ and $[\texttt{count}(m_l)]$. By inductive assumption we already know that h_l satisfies properties 1 to 4 for all $i < l$. Thus, we only need to show them for $i = l$.

Let us show property 1: Let $m_{i_1}, \ldots, m_{i_k}$ with $i_1 < \cdots < i_k < l$ be the children of m_l. Further, if m_l corresponds to the second child of a fork clique, then let m_{i_0} be its sibling. Clearly, $i_0 < i_1$. According to Observation 5.5 max clique A_l is the disjoint union of A'_l and the sets $A_l \cap A_{i_j}$ for $j \in [k]$ if A_l is not the second child of a fork clique, and for $j \in [0, k]$ otherwise. Consequently, $h_l(A_l)$ is the disjoint union of $h_l(A'_l)$ and $h_l(A_l \cap A_{i_j})$ for all feasible $j \leq k$. First, let us consider the children of m_l, that is, all m_{i_j} with $j \in [k]$. For each child m_{i_j} of m_l, we have $A_l \cap A_{i_j} = S^1_{i_j}$. Now suppose for the child m_{i_j}, we have $\texttt{isforkclique}(m_{i_j}) = \text{false}$. Then max clique A_{i_j} is neither a fork clique nor the second child of a fork clique. Therefore, we have $h_l(A_l \cap A_{i_j}) = h_l(S^1_{i_j}) = h_{l-1}(S^1_{i_j}) = [\mathtt{count}(m_{i_j}) - \mathtt{inparent}(m_{i_j}) + 1,\ \mathtt{count}(m_{i_j})]$ by inductive assumption. Next, let us assume $\mathtt{isforkclique}(m_{i_j}) = \text{true}$. Then vertex m_{i_j} corresponds to a fork clique. Let m_i and $m_{i'}$ be the children of the vertex m_{i_j}. Since $m_{i'}$ corresponds to the second child of a fork clique, we know by inductive assumption that $h_l(S^2_{i'}) = h_{l-1}(S^2_{i'}) = [\mathtt{count}(m_{i'}) - \mathtt{inparent}(m_{i'}) + \mathtt{in2children}(m_{i_j}) + 1,\ \mathtt{count}(m_{i'})]$. Further, m_i corresponds neither to a fork clique nor to the second child of a fork clique. Consequently, $h_l(S^1_i) = h_{l-1}(S^1_i) = [\mathtt{count}(m_i) - \mathtt{inparent}(m_i) + 1,\ \mathtt{count}(m_i)]$. The set S^2_i contains exactly the vertices $v \in S^1_i$ with $\#\mathrm{anc}_{A_i}(v) \neq 2$. Therefore, property 2 yields that $h_l(S^2_i) = [\mathtt{count}(m_i) - \mathtt{inparent}(m_i) + \mathtt{in2children}(m_{i_j}) + 1,\ \mathtt{count}(m_i)]$. Clearly, since max clique A_{i_j} is a fork clique, the set $h_l(A_l \cap A_{i_j}) = h_l(S^1_{i_j})$ is the disjoint union of the sets $h_l(S^2_i)$ and $h_l(S^2_{i'})$. Now suppose the vertex m_l corresponds to the second child of a fork clique, and let us consider m_{i_0}, the sibling of m_l. The vertex m_{i_0} corresponds neither to a fork clique nor to the second child of a fork clique. Thus, we have

$$h_l(S^1_{i_0}) = h_{l-1}(S^1_{i_0}) = [\mathtt{count}(m_{i_0}) - \mathtt{inparent}(m_{i_0}) + 1,\ \mathtt{count}(m_{i_0})].$$

The set $A_l \cap A_{i_0}$ contains precisely the vertices $v \in S^1_{i_0}$ with $\#\mathrm{anc}_{A_{i_0}}(v) = 2$, that is, the vertices that are contained in the parent A_{l+1} of the max cliques A_{i_0} and A_l and that are also contained in both of A_{l+1}'s children. As a consequence, property 2 implies that

$$\begin{aligned} h_l(A_l \cap A_{i_0}) = [&\mathtt{count}(m_{i_0}) - \mathtt{inparent}(m_{i_0}) + 1, \\ &\mathtt{count}(m_{i_0}) - \mathtt{inparent}(m_{i_0}) + \mathtt{in2children}(m_{l+1})], \end{aligned}$$

where the vertex m_{l+1} is the parent of the two vertices m_l and m_{i_0}. Finally, by the definition of the mapping h_l we know that $h_l(A'_l) = [\mathtt{count}(m_l) - \mathtt{in0children}(m_l) + 1,\ \mathtt{count}(m_l)]$ if the vertex m_l does not correspond to the second child of a fork clique, and that $h_l(A'_l) = [\mathtt{count}(m_l) - \mathtt{in0children}(m_l) + \mathtt{in2children}(m_{l+1}) + 1,\ \mathtt{count}(m_l)]$ otherwise. Thus, we have shown that the disjoint union of $h_l(A'_l)$ and the sets $h_l(A_l \cap A_{i_j})$ for all feasible $j \leq k$ is exactly the set B_{m_l}. Hence, $h_l(A_l) = B_{m_l}$.

We prove the remaining properties separately for star cliques and for max cliques that are neither star nor fork cliques. We first consider the case where A_l is a star clique. Let us show property 2. We have to prove that $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$ for vertices $v, v' \in A_l$ with $h_l(v) \leq h_l(v')$ if A_l is neither a fork clique nor the second child of a fork clique. Thus, suppose A_l is a star clique that is not the second child of a fork clique. Let $A_{i_1}, \dots, A_{i_k}$ with $i_1 < \cdots < i_k$ be the children of A_l. As shown above A_l is the disjoint union of A'_l and $A_l \cap A_{i_j}$ for all $j \in [k]$. As A_l is a star clique we know $\#\mathrm{anc}_{A_l}(v) = 1$ for all $v \in A_l \cap A_{i_j}$ for $j \in [k]$. Now let us consider $v, v' \in A_l$ with $h_l(v) \leq h_l(v')$. If $v \in A_l \setminus A'_l$ and $v' \in A_l$, we have $\#\mathrm{anc}_{A_l}(v) = 1$ and therefore $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$. It remains to consider the case where $v \in A'_l$. Since $h_l(v) \leq h_l(v')$ and each number in $h(A'_l)$ is greater than every number in $h(A_l \setminus A'_l)$, we also have $v' \in A'_l$. Then $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$ follows directly from the construction of h_l. To show property 3 we suppose again that A_l is a star clique that is not the second child of a fork clique. We have already seen that $\#\mathrm{anc}_{A_l}(v) = 1$ for all $v \in A_l \setminus A'_l$. Therefore, we have $S^1_l \subseteq A'_l$. Now $h_l(S^1_l) = [\mathtt{count}(m_l) - \mathtt{inparent}(m_l) + 1,\ \mathtt{count}(m_l)]$ follows directly from property 2. It remains to show property 4. This time, assume A_l is a star clique that is the second child of a fork clique A_{l+1}. According to Observation 5.5, all vertices in A_l are either contained in a child max clique of A_l, in its sibling max clique, or in A'_l. We know $\#\mathrm{anc}_{A_l}(v) = 1$ for all $v \in A_l$ that are also contained in a child of A_l, and $\#\mathrm{anc}_{A_l}(v) = 2$ for $v \in A_l$ if and only if v is also contained in the sibling max clique of A_l. Consequently, S_l^2 must be a subset of A'_l, and property 2 yields that $h_l(S_l^2) = [\texttt{count}(m_l) - \texttt{inparent}(m_l) + \texttt{in2children}(m_{l+1}) + 1,\ \texttt{count}(m_l)]$.

Now let us consider max cliques A_l that are neither fork cliques nor star cliques. Then A_l cannot be the parent or a child of a fork clique, as the neighbors of fork cliques are star cliques according to Corollary 3.8. Further, A_l must have precisely one child and a parent, since A_l has at most one child by Corollary 4.6 and max cliques of degree 1 are trivially star cliques. To show property 2 let us consider $v, v' \in A_l$ with $h_l(v) \leq h_l(v')$. The child A_{l-1} of max clique A_l is neither a fork clique nor the second child of a fork clique. Thus, according to the inductive assumption we have $\#\mathrm{anc}_{A_{l-1}}(v) \leq \#\mathrm{anc}_{A_{l-1}}(v')$ for $v, v' \in A_{l-1}$. Further, if $v, v' \in A'_l = A_l \setminus A_{l-1}$, then $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$ follows directly from the construction of h_l. Since every number in $h(A'_l)$ is greater than each number in $h(A_l \setminus A'_l)$, it remains to consider v, v' with $v \in A_l \setminus A'_l$ and $v' \in A'_l$. Let us assume that $\#\mathrm{anc}_{A_l}(v) > \#\mathrm{anc}_{A_l}(v')$ for such v and v'. Then $\mathcal{M}_{v'}$ is a separator of the path induced by $\mathcal{M}_v$ in the clique tree of G, which is a contradiction to Corollary 3.3. Thus, $\#\mathrm{anc}_{A_l}(v) \leq \#\mathrm{anc}_{A_l}(v')$ for all $v, v' \in A_l$ with $h_l(v) \leq h_l(v')$. Next, let us show property 3. We know that $S_{l-1}^1 = A_l \cap A_{l-1}$. As A_{l-1} is neither a fork clique nor the second child of a fork clique, we have $h_l(S_{l-1}^1) = [\texttt{count}(m_{l-1}) - \texttt{inparent}(m_{l-1}) + 1,\ \texttt{count}(m_{l-1})]$ by inductive assumption. Further, the set $h_l(A'_l)$ is precisely the interval $[\texttt{count}(m_{l-1}) + 1,\ \texttt{count}(m_l)]$. Hence, $h_l(A_l)$ is the interval $[\texttt{count}(m_{l-1}) - \texttt{inparent}(m_{l-1}) + 1,\ \texttt{count}(m_l)]$, and property 3 follows directly from property 2. Finally, property 4 holds trivially since A_l cannot be the second child of a fork clique. □

Corollary 5.8. *Let $m_1, \ldots, m_{|\mathcal{M}|}$ be the post-order traversal sequence of the underlying directed tree of $K(S_G^R)$. Further, let $B_{m_1}, \ldots, B_{m_{|\mathcal{M}|}}$ be the cliques computed by the algorithm and $A_1, \ldots, A_{|\mathcal{M}|}$ be a transferred traversal sequence. Then there exists a bijection h between V and $[|V|]$, so that for all $i \in [|\mathcal{M}|]$ we have $h(A_i) = B_{m_i}$.*

Let $m_1, \ldots, m_{|\mathcal{M}|}$ be the post-order traversal sequence of the underlying directed tree of $K(S_G^R)$, and let $B_{m_1}, \ldots, B_{m_{|\mathcal{M}|}}$ be the cliques computed by the algorithm. We define the ordered graph $K(G) = (V'_K, E'_K, \leq'_K)$ as follows: We let V'_K be the set $[|V|]$, relation $\leq'_K$ be the natural linear order on $[|V|]$, and we let $\{v, v'\} \in E'_K$ if and only if $v \neq v'$ and there exists an $i \in [|\mathcal{M}|]$ such that $v, v' \in B_{m_i}$.

Corollary 5.9. *The presented algorithm computes the max cliques of the ordered graph $K(G) = (V'_K, E'_K, \leq'_K)$, which is an ordered copy of G.*

Proof. Let $m_1, \ldots, m_{|\mathcal{M}|}$ be the post-order traversal sequence, $B_{m_1}, \ldots, B_{m_{|\mathcal{M}|}}$ be the cliques computed by the algorithm and $A_1, \ldots, A_{|\mathcal{M}|}$ be a transferred traversal sequence. By Corollary 5.8, there exists a bijection h between V and $[|V|]$, so that for all $i \in [|\mathcal{M}|]$ we have $h(A_i) = B_{m_i}$. Then h is a graph isomorphism between G and the graph (V'_K, E'_K), because for all $v, v' \in V$:

There is an edge between v and v' in G.

$\iff$ There exists an $i \in [|\mathcal{M}|]$ such that $v, v' \in A_i$ and $v \neq v'$.

$\iff$ There exists an $i \in [|\mathcal{M}|]$ such that $h(v), h(v') \in B_{m_i}$ and $h(v) \neq h(v')$.

$\iff$ There is an edge between $h(v)$ and $h(v')$ in (V'_K, E'_K).

Consequently, $K(G)$ is an ordered copy of G and the computed cliques $B_{m_1}, \ldots, B_{m_{|\mathcal{M}|}}$ are max cliques. □

Analysis of Space Complexity. Finally we show that the presented algorithm only needs logarithmic space. This finishes the proof of Lemma 5.2, that is, this shows that there exists a logarithmic-space algorithm that, given the canon $K(S_G^R)$ of a supplemented clique tree of a connected chordal claw-free graph G, computes the set of max cliques of an ordered copy $K(G)$ of G.

Proof of Lemma 5.2. According to Corollary 5.9, the presented algorithm computes the max cliques of an ordered copy of G, given the canon $K(S_G^R)$ of a supplemented clique tree of a connected chordal claw-free graph G. It remains to show that the algorithm only needs logarithmic space. In the following, we analyze the space required by the algorithm.

During the depth-first traversal, we need to remember the current vertex, the last move and `count`. As we want to visit the vertices in post-order, we also compute the next move at each vertex. If it is not **down**, then we visit the current vertex for the last time and it belongs to the post-order traversal sequence. Clearly, post-order depth-first traversal is possible in logarithmic space.

At each vertex m, we distinguish between different cases and compute the partial intervals that form B_m. In order to do this, we need the values `in0children`(m'), `inparent`(m'), `in2children`(m'), `isforkclique`(m'), `isforkchild2`(m') and `count`(m') for certain vertices m'. Note that we do not need to remember any of these values. We can recompute them whenever we need them.

For each vertex m', the values `in0children`(m'), `inparent`(m') and `in2children`(m') can be determined in logarithmic space. We obtain these values directly from the color of m'. Further, `isforkclique`(m') can be computed in logarithmic space for every m'. Fork cliques are the only kind of max cliques that contain a vertex which is also contained in (at least) two child max cliques. (Observation 5.5). Thus, we can use the value `in2children`(m') to determine whether a vertex m' corresponds to a fork clique, that is, whether `isforkclique`(m') is true. The value `isforkchild2`(m') can be computed in logarithmic space, by deciding whether m' is the second child of a vertex corresponding to a fork clique.

For every m', we can recompute `count`(m') in logarithmic space by performing a new post-order traversal. Let us look at the value `count` after visiting a vertex m'' during this new post-order traversal: If `isforkclique`(m'') is true, `count` does not change. If `isforkclique`(m'') is false, then depending on the value `isforkchild2`(m''), the value `count` is increased by `in0children`(m'') or by `in0children`$(m'') -$ `in2children`(p'') where p is the parent vertex of m''. Hence, a new post-order traversal allows us to recompute `count`(m').

We can conclude that the presented algorithm only needs logarithmic space. Hence, there is a logarithmic-space algorithm that, given the canon $K(S_G^R)$ of a supplemented clique tree of a connected chordal claw-free graph G, computes the set of max cliques of an ordered copy of G. □

6. Implications

In the previous section, we have shown that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization. This result has interesting consequences for descriptive complexity theory and computational graph theory. We present these consequences in this section.

The following corollary provides a logical characterization of LOGSPACE on the class of chordal claw-free graphs. It is an implication of Theorem 5.1 and Proposition 2.10.

Corollary 6.1. $\mathsf{LREC}_=$ *captures* LOGSPACE *on the class of chordal claw-free graphs.*

Since $\mathsf{LREC}_=$ is contained in $\mathsf{FP{+}C}$ [GGHL12], Theorem 5.1 also implies that there exists an $\mathsf{FP{+}C}$-canonization of the class of chordal claw-free graphs. As a consequence (see [EF99], e.g.), we also obtain a logical characterization of PTIME on the class of chordal claw-free graphs:

Corollary 6.2. $\mathsf{FP{+}C}$ *captures* PTIME *on the class of chordal claw-free graphs.*

Because of $\mathsf{LREC}_=$'s logarithmic-space data complexity, Theorem 5.1 further yields the two subsequent corollaries. These corollaries allow us to reclassify the computational complexity of graph canonization and the graph isomorphism problem on the class of chordal claw-free graphs.

Corollary 6.3. *There exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs.*

Corollary 6.4. *On the class of chordal claw-free graphs, the graph isomorphism problem can be computed in logarithmic space.*

7. Conclusion

Currently, there exist hardly any logical characterizations of LOGSPACE on non-trivial natural classes of unordered structures. The only ones previously presented are that $\mathsf{LREC}_=$ captures LOGSPACE on (directed) trees and interval graphs [GGHL11, GGHL12]. By showing that $\mathsf{LREC}_=$ captures LOGSPACE also on the class of chordal claw-free graphs, we contribute a further characterization of LOGSPACE on an unordered graph class. It would be interesting to investigate further classes of unordered structures such as the class of planar graphs or classes of graphs of bounded treewidth. The author conjectures that $\mathsf{LREC}_=$ captures LOGSPACE on the class of all planar graphs that are equipped with an embedding.

We also make a contribution to the investigation of PTIME's characteristics on restricted classes of graphs. In this paper, we prove that $\mathsf{FP{+}C}$ captures PTIME on the class of chordal claw-free graphs. Thus, the class of chordal claw-free graphs can be added to the (so far) short list of graph classes that are not closed under taking minors and on which PTIME is captured.

Our main result, which states that the class of chordal claw-free graphs admits $\mathsf{LREC}_=$-definable canonization, does not only imply that $\mathsf{LREC}_=$ captures LOGSPACE and $\mathsf{FP{+}C}$ captures PTIME on this graph class, but also that there exists a logarithmic-space canonization algorithm for the class of chordal claw-free graphs. Hence, the isomorphism problem for this graph class is solvable in logarithmic space.

Acknowledgements.

The author wants to thank Nicole Schweikardt and the reviewers for helpful comments that contributed to improving the paper.

References

[BP93] J.R.S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J.R. Gilbert, and J.W.H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 1–29. Springer New York, 1993.

[Bun74] P. Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9:205–212, 1974.

[CFI92] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.

[EF99] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1999.

[EI00] K. Etessami and N. Immerman. Tree canonization and transitive closure. *Information and Computation*, 157(1–2):2–24, 2000.

[Fag74] R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R.M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings 7*, pages 43–73, 1974.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[GGHL11] M. Grohe, B. Grußien, A. Hernich, and B. Laubner. L-recursion and a new logic for logarithmic space. In *CSL*, pages 277–291, 2011.

[GGHL12] M. Grohe, B. Grußien, A. Hernich, and B. Laubner. L-recursion and a new logic for logarithmic space. *Logical Methods in Computer Science*, 9(1), 2012.

[GM99] M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In *Proceedings of the 7th International Conference on Database Theory (ICDT)*, pages 70–82, 1999.

[Gro98] M. Grohe. Fixed-point logics on planar graphs. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 6–15, 1998.

[Gro08] M. Grohe. Definable tree decompositions. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 406–417, 2008.

[Gro10a] M. Grohe. Fixed-point definability and polynomial time on chordal graphs and line graphs. In A. Blass, N. Dershowitz, and W. Reisig, editors, *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, pages 328–353, 2010.

[Gro10b] M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2010.

[Gro13] M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory. `http://lii.rwth-aachen.de/images/Mitarbeiter/pub/grohe/cap/all.pdf`, 2013. [Online; accessed 2017-03-31].

[Gru17a] B. Grußien. Capturing logarithmic space and polynomial time on chordal claw-free graphs. In *Proceedings of the 26th EACSL Annual Conference on Computer Science Logic (CSL)*, pages 26:1–26:19, 2017.

[Gru17b] B. Grußien. *Capturing Polynomial Time and Logarithmic Space using Modular Decompositions and Limited Recursion*. PhD thesis, Humboldt-Universität zu Berlin, 2017.

[Gru17c] B. Grußien. Capturing polynomial time using modular decomposition. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2017.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, 1987.

[Lau10] B. Laubner. Capturing polynomial time on interval graphs. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 199–208, 2010.

[Lau11] B. Laubner. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time*. PhD thesis, Humboldt-Universität zu Berlin, 2011.

[Lin92] S. Lindell. A logspace algorithm for tree canonization. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 400–404, 1992.

[Rei05] O. Reingold. Undirected ST-connectivity in log-space. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 376–385, 2005.

[Sed02] R. Sedgewick. *Algorithms in Java: Parts 1-4*. Addison-Wesley, 3rd edition, 2002.

[Var82] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 137–146, 1982.

[Wal72] J.R. Walter. *Representations of Rigid Cycle Graphs.* PhD thesis, Wayne State University, 1972.