Universiteit Utrecht

Recognizability Equals Definability for k-Outerplanar Graphs

and a Myhill-Nerode Type Proof Technique for Courcelle's Conjecture

by

Lars Jaffke

A thesis submitted in partial fulfillment for the degree of Master of Science

in the
Faculty of Science
Department of Information and Computing Sciences
Supervisor: Prof. dr. Hans L. Bodlaender

July 2015

“Hajo.”

Ein Badner

Abstract

One of the most famous algorithmic meta-theorems states that every graph property that can be defined by a sentence in counting monadic second order logic (CMSOL) can be checked in linear time for graphs of bounded treewidth, which is known as Courcelle's Theorem [9]. These algorithms are constructed as finite state tree automata, and hence every CMSOL-definable graph property is recognizable. Courcelle also conjectured that the converse holds, i.e. every recognizable graph property is definable in CMSOL for graphs of bounded treewidth. We give two types of self-contained proofs of this conjecture for a number of special cases. First, we show that it holds in a stronger form, that is, we prove that recognizability implies MSOL-definability (the counting operation of CMSOL is not needed) for Halin graphs, 3-connected or bounded degree k-outerplanar graphs and some related graph classes. Second, we show that recognizability implies CMSOL-definability for general k-outerplanar graphs.

Acknowledgements

I would like to thank the treewidth club for the interesting talks, the treewidth room for the inspiring and 'gezellige' working environment and toepen (especially the time I beat Frank 0-15). Pizzeria Tricolore for the numerous lunch broodjes (a rough estimate is about 100). Gustav Mahler for his second symphony. Bernhard Mundorf and Fokke Dijkstra. I would like to thank sincerely my friends — for the good times, the distraction, and for not being able to understand what I would have talked about if I had ever dared to try discussing details of this thesis with them in my spare time. And the Unicorns.[(i)]

I would like to extend my heartfelt gratitude to my supervisor, Hans Bodlaender, for the great support, the patience - especially while introducing me to this topic - and for always having had time to answer my questions — whether short or long, whether good or silly. (And also for not pointing out the latter ones.)

Most of all I would like to thank my family; my brother and my parents. For everything.

[(i)] A group of people, not the mythical creatures.

Contents

1

Introduction

In a seminal paper from 1976, Rudolf Halin (1934-2014), lay the ground work for the notion of tree decompositions of graphs [20], which later was studied deeply in the proof of the famous Graph Minor Theorem by Robertson and Seymour [27] and ever since became one of the most important tools for the design of FPT-algorithms for NP-hard problems on graphs. He was also the first one to extensively study the class of planar graphs constructed by a tree and adding a cycle through all its leaves, now known as Halin graphs [19].

Another seminal result is Courcelle's Theorem [9], which states that for every graph property P that can be formulated in a language called counting monadic second order logic (CMSOL), and each fixed k, there is a linear time algorithm that decides P for a graph given a tree decomposition of width at most k (while similar results were discovered by Arnborg et al. [2] and Borie et al. [7]). Counting monadic second order logic generalizes monadic second order logic (MSOL) with a collection of predicates testing the size of sets modulo constants. Courcelle showed that this makes the logic strictly more powerful [9], which can be seen in the following example.

Example 1.1. Let P denote the property that a graph has an even number of vertices. Then P is trivially definable in CMSOL, but it is not in MSOL.

The algorithms constructed in Courcelle's proof have the shape of a finite state tree automaton and hence we can say that CMSOL-definable graph properties are recognizable (or, equivalently, regular or finite-state). Courcelle's Theorem generalizes one direction of a classic result in automata theory by Büchi, which states that a language is recognizable, if and only if it is MSOL-definable [8]. Courcelle conjectured in 1990 that the other direction of Büchi's result can also be generalized for graphs of bounded treewidth in CMSOL, i.e. that each recognizable graph property is CMSOL-definable.

This conjecture is still regarded to be open. Its claimed resolution by Lapoire [24] is not considered to be valid by several experts. In the course of time proofs were given for

the classes of trees and forests [9], partial 2-trees [10], partial 3-trees and k-connected partial k-trees [22]. A sketch of a proof for graphs of pathwidth at most k appeared at ICALP 1997 [21]. Very recently, Bodlaender, Heggernes and Telle gave a proof for partial k-trees without chordless cycles of length at least ℓ [5].

In this thesis we give self-contained proofs of two types of Courcelle's Conjecture. One, where we do not need the counting predicate of CMSOL — among others we prove the conjecture in this somewhat stronger form for Halin graphs and bounded degree or 3-connected k-outerplanar graphs. We give a proof that recognizability implies definability in counting monadic second order logic for all k-outerplanar graphs.

In our proofs, we use another classic result rooted in automata theory, the Myhill-Nerode Theory [25][26]. It states that a language L is recognizable if and only if there exists an equivalence relation $\sim_L$, describing L, that has a finite number of equivalence classes (i.e. $\sim_L$ has *finite index*). Abrahamson and Fellows [1] noted that the Myhill-Nerode Theorem can also be generalized to graphs of bounded treewidth (see also [18, Theorem 12.7.2]): Each graph property P is recognizable if and only if there exists an equivalence relation $\sim_P$ of finite index, describing P, defined over terminal graphs with a bounded number of terminal vertices. This result was recently generalized to hypergraphs [32].

The general outline of our proofs can be described as follows. Given a graph property P, we assume the existence of an equivalence relation $\sim_P$ of finite index. We then show that, given a tree decomposition of bounded width, we can derive the equivalence classes of terminal subgraphs w.r.t. its nodes from the equivalence classes of their children. Once we reach the root of the tree decomposition we can decide whether a graph has property P by the equivalence class its terminal subgraph is contained in. We then show that this construction is MSOL-definable.

The rest of this thesis is organized as follows. In Chapter 2, we give the basic definitions and explain all concepts that we use in more detail. In Chapter 3 we prove some technical results regarding equivalence classes w.r.t. nodes in tree decompositions. The main results are presented in Chapters 4, 5 and 6 which contain proofs of Courcelle's Conjecture for several graph classes (see above). We give some concluding remarks in Chapter 7.

2

Preliminaries

In this chapter we define the basic concepts used in various chapters throughout this thesis. We begin by giving some notational conventions.

Throughout this thesis, $k \in \mathbb{N}$ denotes a constant. Let $a, b \in \mathbb{N}$ with $a < b$. Then, $\mathbb{N}_{|a,b}$ denotes the set $\{a, a+1, \ldots, b\}$ and for a single number $a \in \mathbb{N}$, we let $\mathbb{N}_{|a} = \mathbb{N}_{|1,a}$. Let X denote a set. Then $\mathcal{P}(X)$ denotes the powerset of X, $\mathcal{P}_c(X)$ the set of all subsets of X up to size c and by $\mathcal{P}_c^{\mathcal{M}}(X)$ we denote the set of all multisets over X up to size c.

2.1 Graphs and Tree Decompositions

The reader is assumed to be familiar with the basic notions of graph theory, and is referred to [17, 1.0 - 1.7, 1.10, 3.0 - 3.2, (4.0 - 4.2), 5.0, (5.2), (12.4)] for a complete overview of the necessary background.

A graph $G = (V, E)$ with vertex set V and edge set E is always assumed to be undirected, connected and simple (unless stated otherwise). Let H be a graph. We denote the *subgraph* relation by $G \sqsubseteq H$, and the *proper subgraph* relation by $G \sqsubset H$. For a set $W \subseteq V$, $G[W]$ denotes the *induced subgraph* over $W \subseteq V$ in G, so $G[W] = (W, E \cap (W \times W))$. We call a set $C \subset V$ a *cut* of G, if $G[V \setminus C]$ is disconnected. An *ℓ-cut* of G is a cut of size ℓ. A set $S \subseteq V$ is said to be *incident* to an ℓ-cut C, if $C \subset S$. We call a graph *ℓ-connected*, if it does not contain a cut of size at most $\ell - 1$.

Definition 2.1 ((Planar) Embedding)**.** A drawing of a graph in the plane is called an *embedding*. If no pair of edges in this drawing crosses, then it is called *planar*.

Definition 2.2 (Halin Graph)**.** A graph $G = (V, E)$ is called a *Halin graph*, if it can be formed by a planar embedding of a tree $T = (V, F)$, none of whose vertices has degree two, and a cycle $C = (W, E \setminus F)$ (where $W \subseteq V$) that connects all leaves of the tree such that the embedding stays planar. We refer to T as *the* tree and to C as *the* cycle of G.

Definition 2.3 (k-Outerplanar Graph)**.** Let $G = (V, E)$ be a graph. G is called a *planar graph*, if there exists a planar embedding of G. An embedding of a graph G is 1-*outerplanar*, if it is planar, and all vertices lie on the outer face. For $k \geq 2$, an embedding of a graph G is k-*outerplanar*, if it is planar, and when all vertices on the outer face are deleted, then one obtains a $(k-1)$-outerplanar embedding of the resulting graph. If G admits a k-outerplanar embedding, then it is called a k-*outerplanar graph*.

One can immediately establish a connection between the two graph classes.

Proposition 2.4. *Halin graphs are 2-outerplanar graphs.*

The following definition will play a central role in many proofs of Chapters 4 and 6.

Definition 2.5 (Fundamental Cycle)**.** Let $G = (V, E)$ be a graph with maximal spanning forest $T = (V, F)$. Given an edge $e = \{v, w\}$, $e \in E \setminus F$, its *fundamental cycle* is a cycle that is formed by the unique path from v to w in F together with the edge e.

We now turn to the notion of tree decompositions and some related concepts.

Definition 2.6 (Tree Decomposition, Treewidth, Partial k-Tree)**.** A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, X) of a tree $T = (N, F)$ and an indexed family of vertex sets $(X_t)_{t \in N}$ (called *bags*), such that the following properties hold.

(i) Each vertex $v \in V$ is contained in at least one bag.

(ii) For each edge $e \in E$ there exists a bag containing both endpoints.

(iii) For each vertex $v \in V$, the bags in the tree decomposition that contain v form a subtree of T.

The *width* of a tree decomposition is the size of the largest bag minus 1 and the *treewidth* of a graph is the minimum width of all its tree decompositions. We sometimes refer to a graph of treewidth at most k as a *partial k-tree*.[(i)]

To avoid confusion, in the following we will refer to elements of N as *nodes* and elements of V as *vertices*. Sometimes, to shorten the notation, we might not differ between the terms *node* and *bag* in a tree decomposition.

Definition 2.7 (Node Types)**.** We distinguish three types of nodes in a tree decomposition (T, X), listed below.

(i) The nodes corresponding to leaves in T are called *leaf* nodes.

[(i)] For a discussion of different characterizations related to the treewidth of a graph, see [4, Section 2].

(ii) If a node has exactly one child it is called an *intermediate* node.

(iii) If a node has more than one child it is called a *branch* node.

As we will typically speak of some direction between nodes in tree decompositions, such as a parent-child relation, we define the following.

Definition 2.8 (Rooted and Ordered Tree Decomposition)**.** Let $(T = (N, F), X)$ be a tree decomposition. We call (T, X) *rooted*, if there is one distinguished node $r \in N$, called the *root* of T, inducing a parent-child relation on all edges in F. If there exists a fixed ordering on all bags sharing the same parent node, then T is called *ordered*.

We now introduce *terminal graphs*, over which we will later define equivalence relations for graph properties.

Definition 2.9 (Terminal Graph)**.** A *terminal graph* $G = (V, E, X)$ is a graph with vertex set V, edge set E and an ordered terminal set $X \subseteq V$. If $|X| = t$, we call G a *t-terminal graph*.

Terminal graphs of special interest in the rest of this thesis are *terminal subgraphs* w.r.t. bags in a tree decomposition. We require the notion of *partial* terminal subgraphs in the proofs of Chapter 3 and Section 5.1.

Definition 2.10 ((Partial) Terminal Subgraph)**.** Let $(T = (N, F), X)$ be a rooted (and ordered) tree decomposition of a graph $G = (V, E)$ with bags X_t and $X_{t'}$, $t, t' \in N$, such that t is the parent node of t'. The graphs defined below are induced subgraphs of G given the respective vertex sets.

(i) A *terminal subgraph* of a bag X_t, denoted by $[X_t]^+$, is a terminal graph induced by the vertices in X_t and all its descendants, with the set X_t as its terminals.

(ii) A *partial terminal subgraph* of X_t *given a child* $X_{t'}$, denoted by $[X_t]^+_{X_{t'}}$ is the terminal graph induced by X_t and the vertices and edges of all terminal subgraphs of the children of X_t that are left siblings of $X_{t'}$, with terminal set X_t.

The ordering in each terminal set of the above mentioned terminal graphs can be arbitrary, but fixed.[(ii)]

Note that in later chapters, one might need to fix an ordering on the terminals of a (partial) terminal subgraph. If the ordering does not have to have any special properties, one can use an ordering based on finding a $(k+1)$-vertex coloring of the graph, which we will define below.

[(ii)] For an illustration of Definition 2.10, see Figure 3.1a, where $H = [X_H]^+$ and $G = [X_G]^+_{X_H}$.

Definition 2.11 (k-Coloring Order)**.** Let $G = (V, E)$ be a graph and (T, X) a tree decomposition of G of width $(k-1)$. Suppose we have a k-coloring $\gamma : V \to \{1, \ldots, k\}$ of G such that in each bag X_t, $t \in N$, each vertex has a different color. Then, the order of the vertices in X_t for each $t \in N$, induced by the coloring γ, is called the *k-coloring order of G for (T, X)*.

One can easily prove the following.

Proposition 2.12. *Let $G = (V, E)$ be a graph and (T, X) a width-k tree decomposition of G. Then, there exists a $(k+1)$-coloring order of G for (T, X).*

We use the following notation. If P denotes a graph property (e.g. a graph contains a Hamiltonian cycle), then by '$P(G)$' we express that a graph G has property P.

2.2 Tree Automata for Graphs of Bounded Treewidth

We briefly review the concept of tree automata and recognizability of graph properties for graphs of bounded treewidth. For an introduction to the topic we refer to [18, Chapter 12]. For the formal details of the following notions, the reader is referred to [22].

A tree automaton $\mathcal{A}$ is a finite state machine accepting as an input a tree structure over an alphabet Σ as opposed to words in classical word automata. Formally, $\mathcal{A}$ is a triple $(\mathcal{Q}, \mathcal{Q}_{Acc}, f)$ of a set of states $\mathcal{Q}$, a set of accepting states $\mathcal{Q}_{Acc} \subseteq \mathcal{Q}$ and a transition function f, deriving the state of a node in the input tree $\mathcal{T}$ from the states of its children and its own symbol $s \in \Sigma$. $\mathcal{T}$ is *accepted* by $\mathcal{A}$, if the state of the root node of $\mathcal{T}$ is an element of the accepting states $\mathcal{Q}_{Acc}$ (after a run of $\mathcal{A}$ with $\mathcal{T}$ as an input).

To recognize a graph property on graphs of treewidth at most k, one encodes a rooted width-k tree decompositions as a labeled tree over a special type of alphabet, in the following denoted by Σ_k (see Definition 3.5, Proposition 3.6 in [22]). We say that a tree automaton over such an alphabet *processes* width-k tree decompositions.

Definition 2.13 (Recognizable Graph Properties)**.** Let P denote a graph property. We call P *recognizable* (for graphs of treewidth k), if there exists a tree automaton $\mathcal{A}_P$ processing width-k tree decompositions, such that following are equivalent.

(i) (T, X) is a width-k tree decomposition of a graph G with $P(G)$.

(ii) $\mathcal{A}_P$ accepts (the labeled tree over Σ_k corresponding to) (T, X).

Note that for an alternative proof of Courcelle's Conjecture (i.e. opposed to our method, see Sections 2.3, 2.4 and 2.5, Conjectures 2.24 and 2.25) using the notions of tree automata processing bounded width tree decompositions directly, one can use Proposition 5.4 in [9]. (See also Lemma 5.4 in [22].)

2.3 Equivalence Relations

In this section we define equivalence relations over terminal graphs (Definition 2.9) and introduce a Myhill-Nerode analog for graphs of treewidth at most k. Intuitively speaking it says that each state in a tree automaton (see the previous section) can be identified with an equivalence class of an equivalence relation, which can be defined for graph properties P.

Definition 2.14 (Gluing via $\oplus$)**.** Let $G = (V_G, E_G, X_G)$ and $H = (V_H, E_H, X_H)$ be two terminal graphs with $|X_G| = |X_H|$. The graph $G \oplus H$ is obtained by taking the disjoint union of G and H and for each i, $1 \leq i \leq |X_G|$, identifying the i-th vertex in X_G with the i-th vertex in X_H.

Note that if an edge is included both in G and in H, we drop one of the edges in $G \oplus H$, i.e. we do not have parallel edges in the graph.

We use the operator $\oplus$ to define equivalence relations over terminal graphs. Throughout this thesis we will restrict ourselves to (at most) t-terminal graphs for some fixed $t \in \mathbb{N}$, since we focus on equivalence relations with a finite number of equivalence classes. These, in general, do not exist for classes of terminal graphs with arbitrary boundary size (see [1]). We furthermore only consider terminal graphs of treewidth (at most) k with at most $k+1$ terminals, as we can disregard all terminal graphs that do not uphold the treewidth bound under consideration. For a discussion of these notions see e.g. [18, Section 12.7].

Definition 2.15 (Equivalence Relation over Terminal Graphs)**.** Let P denote a graph property and let all (terminal) graphs below be of treewidth k. $\sim_{P_t}$ denotes the equivalence relation over t-terminal graphs of treewidth k, where $1 \leq t \leq k+1$, defined as follows. Let G, H and K be t-terminal graphs. Then we have:

$$G \sim_{P_t} H \Leftrightarrow \forall K : P(G \oplus K) \Leftrightarrow P(H \oplus K)$$

For terminal graphs with *at most* $k+1$ terminals of treewidth k, we furthermore define the equivalence relation $\sim_{P_{\leq k}}$. Let $1 \leq t \leq k+1$.

$$G \sim_{P_{\leq k}} H \Leftrightarrow |X_G| = |X_H| \ (= t) \wedge G \sim_{P_t} H$$

This yields notions of *equivalence classes* and *finite index* (for both $\sim_{P_t}$ and $\sim_{P_{\leq k}}$) in the ordinary way. We might drop the index of $\sim_{P_{\leq k}}$ and refer to it as $\sim_P$ or simply $\sim$, if it is clear from the context.

We illustrate Definition 2.15 with an example.

Example 2.16. Let P denote the property that a graph has a Hamiltonian cycle. Let G and H be two terminal graphs (of bounded treewidth) with terminal sets X_G and X_H, respectively (where $|X_G| = |X_H| = t$). We say that G and H are equivalent w.r.t. $\sim_{P_t}$, if for all terminal graphs K (with terminal set X_K, $|X_K| = t$), the graph $G \oplus K$ contains a Hamiltonian cycle if and only if $H \oplus K$ contains a Hamiltonian cycle. A simple case when this hols is when both G and H contain a Hamiltonian path such that their terminal sets consist of the two endpoints of the path.

Definition 2.17 (P-Equivalence Class)**.** Let P denote a graph property and C an equivalence class of $\sim_{P_{\leq k}}$. We call C a *P-equivalence class*, if the following holds.

$$\forall G \in C : P(G \oplus (X_G, \emptyset, X_G))$$

As mentioned earlier, our proof technique for Courcelle's Conjecture is based on the Myhill-Nerode Theory for graphs of treewidth at most k. The following theorem formally states this result.

Theorem 2.18 (Myhill-Nerode Analog for Graphs of Treewidth at most k (cf. Theorem 12.7.2 in [18]))**.** *Let P denote a graph property. Then the following are equivalent for any fixed k.*

(i) P is recognizable for graphs of treewidth at most k.

(ii) $\sim_{P_{\leq k}}$ has finite index.

By the proof of this theorem (see [18, p. 254 ff.]) we know that we can identify the P-equivalence classes of $\sim_{P_{\leq k}}$ with the accepting states in the corresponding automaton. We will use this fact in the proofs of Sections 4.3 and 5.1.

2.4 Monadic Second Order Logic of Graphs

We now define counting monadic second order logic of graphs $G = (V, E)$, using terminology from [7] and [22]. Variables in this predicate logic are either single vertices/edges or vertex/edge sets. We form predicates by joining *atomic predicates* (vertex equality $v = w$, vertex membership $v \in V$, edge membership $e \in E$ and vertex-edge incidence $\mathrm{Inc}(v, e)$) via negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and equivalence $\leftrightarrow$

together with existential quantification $\exists$ and universal quantification $\forall$ over variables in our domain $V \cup E$. To extend this monadic second order logic (MSOL) to *counting* monadic second order logic (CMSOL), one additionally allows the use of predicates $\mathrm{mod}_{p,q}(S)$ for sets S, which are true, if and only if $|S| \bmod q = p$, for constants p and q (with $p < q$).

Let ϕ denote a predicate without unquantified (so-called *free*) variables constructed as explained above and G be a graph. We call ϕ a *sentence* and denote by $G \models \phi$ that ϕ yields a truth assignment when evaluated with the graph G.

Definition 2.19 (Definable Graph Properties)**.** Let P denote a graph property. We say that P is *(C)MSOL-definable*, if there exists a (C)MSOL-sentence ϕ_P such that $P(G)$ if and only if $G \models \phi_P$.

A predicate ϕ can have two types of free variables. The first one is a number of *arguments* $x_1, \ldots, x_a$ and we denote our predicate by $\phi(x_1, \ldots, x_a)$. Predicates with arguments are used to define relations in (C)MSOL and typically appear as sub-predicates in more complex statements defining a graph property. Secondly, a predicate can have a number of *parameters*, which can be seen as auxiliary variables to define a graph property in (C)MSOL and do not appear in the notation of a predicate.

Example 2.20. Let P denote the property that a graph has a k-coloring and $\phi_{col}(v, w)$ a predicate, which is true, if and only if a vertex v has a lower numbered color than w in a given coloring. Then ϕ_{col} has two arguments, vertices v and w, and k parameters, the k color classes. Clearly, the choice of the parameters influences the evaluation of ϕ_{col}, but in most applications of parameters for predicates, it is sufficient to show that one can guess *some* variables of the evaluation graph to define a property (or a relation).

We introduce another term of definability based on predicates that have arguments and/or parameters.

Definition 2.21 (Existential Definability)**.** Let $R(x_1, \ldots, x_r)$ denote a relation with arguments $x_1, \ldots, x_r$. We say that R is *(C)MSOL-definable*, if there exists a parameter-free predicate $\phi_R(x_1, \ldots, x_r)$, encoding the relation R. Furthermore we call R *existentially (CMSOL)-definable*, if there exists a predicate $\phi_R(x_1, \ldots, x_r)$ with parameters $x_1, \ldots, x_p$, which, after substituting the parameters by fixed values of an evaluation graph, encodes the relation R. For a graph property P and an argument-free predicate ϕ_P with parameters $x_1, \ldots, x_p$, we define the term existential (C)MSOL-definability analogously.[(iii)]

A central concept used in this thesis is an implicit representation of a tree decomposition in monadic second order logic, as we cannot refer to its bags and edges as variables in MSOL directly. Hence, we require two types of predicates.

[(iii)] Note that in the more informal parts of this thesis, we might not always differ between the terms 'definability' and 'existential definability'.

The first one will allow us to verify whether a vertex is contained in some bag and whether any vertex set in the graph constitutes a bag in its tree decomposition. In our definition, each bag will be associated with either a vertex or an edge in the underlying graph together with a *type*. A type usually defines a set of vertices that have a certain (typically structural) relation to the vertex or edge, to which the bag corresponds. In the definition below we denote these types by τ_i (for vertex types) and σ_j (for edge types), which up to now do not have any specific meaning. It is important to note, however, that the number of these types has to be constant.

The second one allows for identifying edges in the tree decomposition, i.e. for any two vertex sets X and Y, this predicate will be true if and only if both X and Y are bags in the tree decomposition and X is the bag corresponding to the parent node of Y.

Definition 2.22 (MSOL-definable tree decomposition)**.** A rooted (and ordered) tree decomposition (T, X) of a graph G is called *existentially MSOL-definable*, if the following are existentially MSOL-definable (with parameters $x_1, \ldots, x_p$ for some constant p).

(i) Each bag X in the tree decomposition can be identified by one of the following predicates (where s and t are constants).

  (a) $\mathrm{Bag}^V_{\tau_1}(v, X), \ldots, \mathrm{Bag}^V_{\tau_t}(v, X)$: The bag X is associated with type τ_i and the vertex $v \in V$, where $1 \leq i \leq t$.

  (b) $\mathrm{Bag}^E_{\sigma_1}(e, X), \ldots, \mathrm{Bag}^E_{\sigma_s}(e, X)$: The bag X is associated with type σ_j and the edge $e \in E$, where $1 \leq j \leq s$.

(ii) There exists a predicate $\mathrm{Parent}(X_p, X_c)$ to identify edges in T, which is true, if and only if X_p is the parent bag of X_c.

We call an existentially MSOL-definable tree decomposition *ordered*, if the following is existentially MSOL-definable, using (some of) the p parameters.

(iii) There exists a predicate $\mathrm{nb}_\prec(X_l, X_r)$, which is true if and only if X_l and X_r are siblings such that X_l is the direct left sibling of X_r.

We can now show that if we have an existentially MSOL-definable tree decomposition of width k for a graph class $\mathcal{C}$, one can write a parameter-free predicate in monadic second order logic, which encodes a width-k tree decomposition of an evaluation graph $G \in \mathcal{C}$, after replacing the parameters with fixed values of G.

Lemma 2.23. *Let (T, X) be an existentially MSOL-definable tree decomposition with parameters $x_1, \ldots, x_p$. There exists a predicate ϕ with zero parameters and p arguments, which is true if and only if the predicates $Bag_{\tau_1}, \ldots, Bag_{\tau_t}$, $Bag_{\sigma_1}, \ldots, Bag_{\sigma_s}$ and Parent describe a width-k rooted tree decomposition of an evaluation graph G.*

Proof. The proof can be done analogously to the proof of Lemma 4.7 in [22]. □

2.5 Courcelle's Conjecture

To conclude this chapter, we will now state Courcelle's Conjecture formally.

Conjecture 2.24 (Courcelle, 1990). Let P denote a graph property. For any fixed $k \in \mathbb{N}$, the following holds. If P is recognized by a finite state tree automaton $\mathcal{A}$ for graphs of treewidth k, then there exists a (C)MSOL-sentence Φ, such that $G \models \Phi$ if and only if $\mathcal{A}$ accepts (T, X), a width-k tree decomposition of G.

By Theorem 2.18, we can reformulate this conjecture.

Conjecture 2.25. Let P denote a graph property. For any fixed $k \in \mathbb{N}$, the following holds. If $\sim_{P_{\leq k}}$ has finite index, then there is a (C)MSOL-sentence Φ, such that $G \models \Phi$, if and only if the terminal subgraph of the root of a width-k tree decomposition (T, X) of G is contained in a P-equivalence class of $\sim_{P_{\leq k}}$.

Throughout this text we will abbreviate Conjecture 2.24 to

'recognizability implies (C)MSOL-definability for graphs of treewidth at most k'

and Conjecture 2.25 to

'finite index implies (C)MSOL-definability for graphs of treewidth at most k'.

3

Deriving Equivalence Class Membership

The current chapter contains a number of technical results related to equivalence classes of (partial) terminal subgraphs of bags in a tree decomposition. In particular, we will show how to derive the equivalence classes of (partial) terminal subgraphs of bags in a tree decomposition from the equivalence classes of some (partial) terminal subgraphs of child/sibling bags. Hence we prove that these equivalence classes are related to each other in the same way as states in some finite automaton via its transition function, which will be of vital importance in the proofs of Sections 4.3 and 5.1.

Note that in the proofs of this section, the terminal sets in all (partial) terminal subgraphs w.r.t. bags in a width-k tree decomposition have to have an ordering on its terminal sets. Since we do not require this ordering to have any specific properties, we will assume that they are ordered according to the $(k+1)$-coloring order (see Definition 2.11, Proposition 2.12).

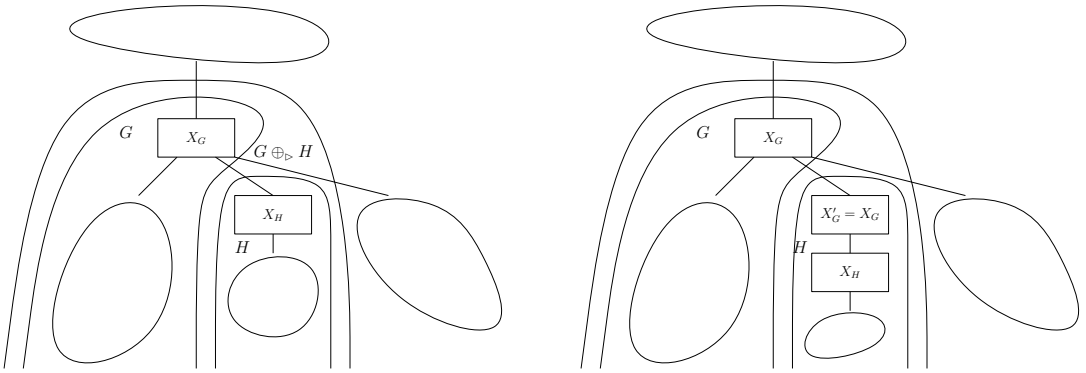
In the following, unless stated otherwise, we assume that a width-k tree decomposition is rooted and ordered. First, we consider branch nodes. We begin by defining an operator, which can be seen as an extension of the $\oplus$-operator.

Definition 3.1 (Gluing via $\oplus_{\triangleright}$)**.** Let X_G be a branch bag in a width-k tree decomposition with child bag X_H and let $G = [X_G]^+_{X_H} = (V_G, E_G, X_G)$ and $H = [X_H]^+ = (V_H, E_H, X_H)$ denote the partial terminal subgraph of X_G given X_H and the terminal subgraph of X_H, respectively. The operation $\oplus_{\triangleright}$ is defined as:

$$G \oplus_{\triangleright} H = (V_G \cup V_H, E_G \cup E_H, X_G)$$

Note that again, we drop parallel edges, if they occur.

Consider the situation depicted in Figure 3.1 and suppose that we know the equivalence class for the graph G and the equivalence class for graph H. We want to derive the



(a) The respective terminal graphs (b) Splitting $\{X_G, X_H\}$

Figure 3.1: Branch node in a tree decomposition

equivalence class of the partial terminal subgraph of X_G given the right sibling of X_H (which is the terminal graph $G \oplus_\triangleright H$).

We will prove that the equivalence class of $G \oplus_\triangleright H$ only depends on the equivalence class of G and H by explaining how we can create a terminal graph in this class from any pair of graphs $G' \sim G$, $H' \sim H$ with $X_{G'} = X_G$ and $X_{H'} = X_H$. Note that since we are only interested in determining whether the underlying graph of the tree decomposition, say G^*, has property P, it is sufficient to only consider terminal graphs in the equivalence classes of G and H that have the same terminal sets as G and H. These classes contain any number of (terminal) graphs, which are completely unrelated to G^* and hence can be disregarded. The following lemma formalizes the above discussion.

Lemma 3.2. *Let X_G be a branch bag in a width-k tree decomposition and X_H one of its child bags. Let $G = [X_G]^+_{X_H}$, $H = [X_H]^+$ and G' and H' two terminal graphs. If $G' \sim G$, $H' \sim H$, $X_G = X_{G'}$ and $X_H = X_{H'}$, then $(G \oplus_\triangleright H) \sim (G' \oplus_\triangleright H')$.*

Proof. We first define an operator that allows us to rewrite $\oplus_\triangleright$.

Definition 3.3 (Gluing via $\oplus_T$)**.** Let G be a (terminal) graph and X an ordered set of vertices. The operation $\oplus_T$ is defined as taking the (not necessarily disjoint) union of X and the vertices in G and let X be the terminal set of the resulting terminal graph, i.e.:

$$G \oplus_T X = (V_G \cup X, E_G, X)$$

Note that $\oplus_T$ can either be used to make a graph a terminal graph, or to equip a terminal graph with a new terminal set. One easily observes the following.

Proposition 3.4. *Let G and H be two terminal graphs as in Lemma 3.2. Then,*

$$G \oplus_\triangleright H = \overbrace{(G \oplus \underbrace{(H \oplus_T X_G)}_{(b)}) \oplus_T X_G}^{(a)}. \tag{3.1}$$

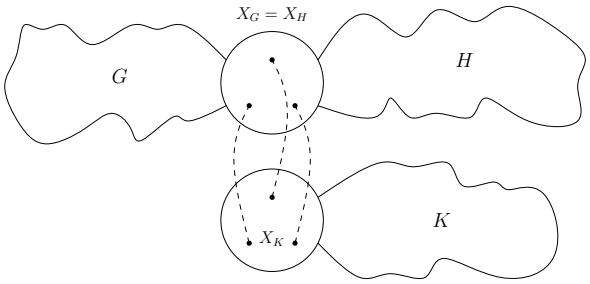


Figure 3.2: Terminal graphs G, H and K as in the proof of Proposition 3.5. The dashed lines indicate, which vertices are being identified in the corresponding $\oplus$-operation.

This process of rewriting $\oplus_{\triangleright}$ can be illustrated as shown in Figure 3.1b. Instead of computing $G \oplus_{\triangleright} H$ directly, we split the edge between the bags X_G and X_H, creating a new bag X'_G in between the edge, where $X'_G = X_G$. Then we extend H to a terminal graph with terminal set X'_G by using the $\oplus_T$-operator. Denote this graph by $H_{X'_G}$. Since $H_{X'_G}$ has terminal set $X'_G = X_G$, we can apply $\oplus$ to G and H', such that all vertices that are identified in the operation are equal. This results in the graph consisting of all vertices and edges in both G and H. Eventually, we apply $\oplus_T$ to the resulting graph again to make it a terminal graph with terminal set X_G.

We will lead the proof of Lemma 3.2 in two steps: First we show that we can construct graphs equivalent to $(G \oplus H) \oplus_T X_G$ by members of the equivalence classes of G and H, if G and H have the same terminal set (Part (a) of Equation 3.1, where H denotes the terminal graph $H \oplus_T X_G$). In the second step, we show that we can construct graphs equivalent to $H \oplus_T X$ from members of the equivalence class H for any terminal set X (Part (b) of Equation 3.1).

Proposition 3.5. *Let $G = (V_G, E_G, X_G)$ and $H = (V_H, E_H, X_H)$ be two terminal graphs with $X_G = X_H$. Let G' and H' be two terminal graphs with $G' \sim G$, $H' \sim H$, $X_G = X_{G'}$ and $X_H = X_{H'}$. Then,*

$$(G \oplus H) \oplus_T X_G \sim (G' \oplus H') \oplus_T X_{G'}.$$

Proof. By Figure 3.2 we can observe the following.

$$K \oplus ((G \oplus H) \oplus_T X_G) = G \oplus ((K \oplus H) \oplus_T X_G)$$

Regardless of the order in which we apply the operators, both graphs will have the same vertex and edge sets. As for the identifying step (using the $\oplus$-operator), one can see that for all $i = 1, \ldots, |X_K|$ we have that the i-th vertex in X_K is identified with the i-th vertex in X_G in the left-hand side of the equation and with the i-th vertex in X_H in the right-hand side. The equality still holds, since $X_G = X_H$. We use this argument (and

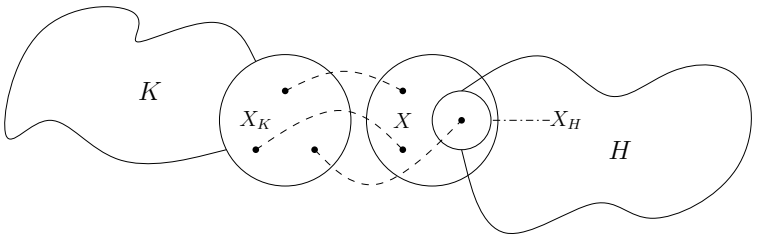


Figure 3.3: Terminal graphs H and K, and a terminal set X. The dashed lines indicate, which vertices are being identified in the corresponding $\oplus$-operation.

the fact that $X_{G'} = X_G = X_H = X_{H'}$) to show the following.

$$\begin{aligned}\forall K:\ & P(K \oplus ((G \oplus H) \oplus_T X_G)) \Leftrightarrow P(G \oplus ((K \oplus H) \oplus_T X_G)) \\ \Leftrightarrow & P(G' \oplus ((K \oplus H) \oplus_T X_{G'})) \Leftrightarrow P(H \oplus ((K \oplus G') \oplus_T X_H)) \\ \Leftrightarrow & P(H' \oplus ((K \oplus G') \oplus_T X_{H'})) \Leftrightarrow P(K \oplus ((G' \oplus H') \oplus_T X_{G'}))\end{aligned}$$

Hence, our claim follows. □

Lemma 3.6. *Let H, H' be terminal graphs with $H \sim H'$, $X_H = X_{H'}$ and X an ordered vertex set. Then, $H \oplus_T X \sim H' \oplus_T X$.*

Proof. By Figure 3.3, one can derive a similar argument as in the proof of Proposition 3.5. Note that $|X_K| = |X|$ (otherwise, $\oplus$ is not defined) and let $K_X = K \oplus (X, \emptyset, X)$, i.e. the graph obtained by identifying each i-th vertex in X_K with each i-th vertex in X, where $1 \leq i \leq |X_K|$. Then,

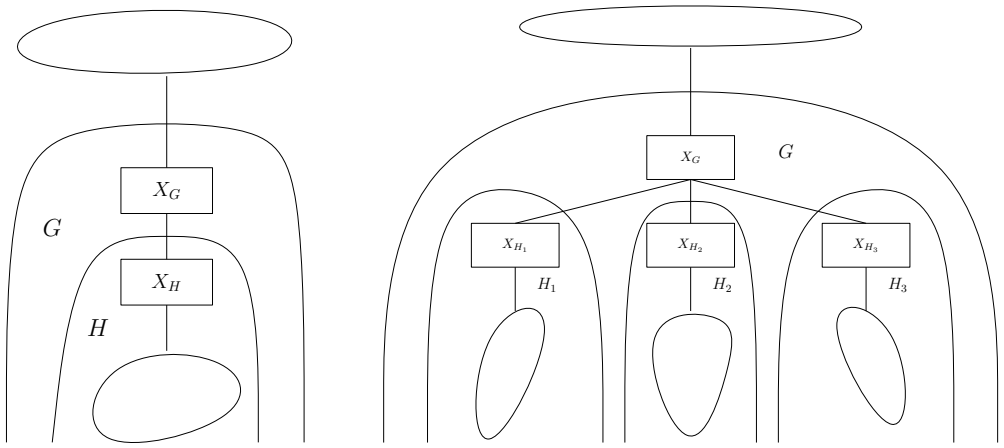
$$K \oplus (H \oplus_T X) = H \oplus (K_X \oplus_T X_H).$$

In the left-hand side, we first extend the terminal graph H to have terminal set X and then glue the resulting graph to K. Thus the i-th vertex in X_K is identified with the i-th vertex in X, $i = 1, \ldots, |X_K|$. The same vertices are being identified in the first step in computing the right-hand side, which is constructing the graph K_X. We then extend this graph to have terminal set X_H and glue it to the graph H. Since again, in both of the computations the same vertices get identified and both graphs have equal vertex and edge sets, we see that our claim holds. We use this argument (and the fact that $X_H = X_{H'}$) to conclude our proof as follows.

$$\begin{aligned}\forall K:\ & P(K \oplus (H \oplus_T X)) \Leftrightarrow P(H \oplus (K_X \oplus_T X_H)) \\ \Leftrightarrow & P(H' \oplus (K_X \oplus_T X_{H'})) \Leftrightarrow P(K \oplus (H' \oplus_T X))\end{aligned}$$

□

This concludes the proof of Lemma 3.2. □



(a) Intermed. node. $G = H \oplus_T X_G$.

(b) Bounded degree branch node. Note that $G = (H_1 \oplus_T X_G) \oplus_\triangleright (H_2 \oplus_T X_G) \oplus_\triangleright (H_3 \oplus_T X_G)$.

Figure 3.4: Intermediate and bounded degree branch node (both X_G in the respective figure) in a tree decomposition.

The methods used in this proof also allow us to handle intermediate nodes in a tree decomposition. For an illustration see Figure 3.4a. Lemma 3.6 suffices as an argument that we can derive the equivalence class of G from graphs equivalent to H.

Next, we generalize the situation of Lemma 3.2, where we were dealing with two child nodes of a branch bag, to handle any constant number of children at a time (see Figure 3.4b). We will apply this result to tree decompositions that are not ordered but instead have bounded degree.

Lemma 3.7. *Let X_G be a branch bag in a width-k tree decomposition with children $X_{H_1}, \ldots, X_{H_c}$ (for a fixed c). Let $H_1 = [X_{H_1}]^+, \ldots, H_c = [X_c]^+$. If $H'_1 \sim H_1, \ldots, H'_c \sim H_c$ and $X_{H'_1} = X_{H_1}, \ldots, X_{H'_c} = X_{H_c}$, then*

$$(H_1 \oplus_T X_G) \oplus_\triangleright \cdots \oplus_\triangleright (H_c \oplus_T X_G) \sim (H'_1 \oplus_T X_G) \oplus_\triangleright \cdots \oplus_\triangleright (H'_c \oplus_T X_G)$$

Proof. Let G and H be the two terminal graphs as indicated below.

$$\underbrace{(H_1 \oplus_T X_G)}_{G} \oplus_\triangleright \underbrace{(H_2 \oplus_T X_G) \oplus_\triangleright \cdots \oplus_\triangleright (H_c \oplus_T X_G)}_{H}$$

Since $H_1 \sim H'_1$, we know by Lemma 3.6 that $(H_1 \oplus_T X_G) \sim (H'_1 \oplus_T X_G)$. Let $G' = (H'_1 \oplus_T X_G)$, then we have that $G \sim G'$. Now, by Lemma 3.2, we know that $(G \oplus_\triangleright H) \sim (G' \oplus_\triangleright H)$ and hence:

$$G \oplus_\triangleright H \sim (H'_1 \oplus_T X_G) \oplus_\triangleright H$$

We can apply this argument repeatedly and our claim follows. Note that the child bags $X_{H_1}, \ldots, X_{H_c}$ do not need a specific ordering, as in this context the operation $\oplus_\triangleright$ is commutative (all graphs, which it is applied to, have terminal set X_G). □

4

Halin Graphs

This chapter is devoted to proving our first main result, which is that MSOL-definability equals recognizability for the class of Halin graphs. As outlined before, we will prove that finite index implies MSOL-definability. In a first step, we will show that we can define a certain orientation on the edges of a Halin graph together with an ordering on edges with the same head vertex in monadic second order logic (Section 4.1), which we then will use to construct MSOL-definable tree decompositions of Halin graphs (Section 4.2). We conclude the proof in Section 4.3.

In many of the proofs of MSOL-definability of graph properties (or properties of tree decomposition), we use other MSOL-predicates we defined at an earlier stage, and refer for more precise expressions to the appendix.

4.1 Edge Orientation and Ordering

In the following we will develop an orientation on the edges of a Halin graph, together with an ordering on edges with the same head vertex, which is MSOL-definable. Our goal is that in this orientation, the edges that form the cycle connecting the leaves is a directed cycle and the tree of the Halin graph forms a directed tree with some arbitrary root on the outer cycle.

Lemma 4.1 (Cf. [12], Lemma 4.8 in [22])**.** *Let G be a graph of treewidth k. Any orientation on its edges using predicates head(e, v) and tail(e, v), which is existentially encoded by a predicate ϕ_{Ori} with p parameters, is existentially MSOL-definable with $p + k + 2$ parameters.*

Proof. Since G has treewidth k, we know that it admits a $k+1$-coloring on its vertices. We assume we are given such a coloring and denote the color set by $\{0, 1, \ldots, k\}$. Now let F be a set of edges of G and $e = \{v, w\}$ an edge in the graph. We know that

$\text{col}(v) \neq \text{col}(w)$ and thus we either have $\text{col}(v) < \text{col}(w)$ or $\text{col}(v) > \text{col}(w)$. We let the edge e be directed from v to w, if

(i) $\text{col}(v) < \text{col}(w)$ and $e \in F$, or

(ii) $\text{col}(v) > \text{col}(w)$ and $e \notin F$

and otherwise from w to v. Thus we can choose any orientation of the edge set of G by choosing the corresponding set F. Assuming that ϕ_{Ori} uses predicates $\text{head}(e, v)$ and $\text{tail}(e, v)$ as shown in Appendix A.1, we can define our predicate as

$$\exists X_1 \cdots \exists X_{k+1}(\exists F \subseteq E)(k+1\text{-col}(V, X_1, \ldots, X_{k+1}) \wedge \phi_{Ori}).$$

The parameters are the color sets $X_1, \ldots, X_{k+1}$ and the edge set F. □

Proposition 4.2. *Let $G = (V, E)$ be a Halin graph. The orientation on the edge set of G such that its tree forms a rooted directed tree and the outer cycle is a directed cycle, is existentially MSOL-definable with 7 parameters.*

Proof. Since Halin graphs have treewidth 3, we can use Lemma 4.1. Let E_T denote the edges in the spanning tree and E_C the edges on the outer cycle. The orientation stated above can be defined in MSOL as

$$\phi_{Ori} = \exists E_T \exists E_C(\text{Part}_E(E, E_T, E_C) \wedge \text{Tree}_{\rightarrow}(V, E_T) \wedge \text{Cycle}_{\rightarrow}(\text{IncV}(E_C), E_C)).$$

We have five parameters for the edge orientation plus the parameters for the edge set of the tree and the cycle. The MSOL-predicates given in Appendix A.1 complete the proof. □

Next, we define an ordering on all edges with the same head vertex in a Halin graph, which we can define in monadic second order logic using the orientation of the edges given above and its fundamental cycles. This is a central step in our proof, as it allows us to avoid using the counting predicate in the construction of our tree decomposition. The main idea in the proof of Lemma 4.3 is that we can order the child edges of a vertex in the order in which their leaf descendants appear on the outer cycle.

Lemma 4.3. *For any vertex in a Halin graph there exists an ordering $nb_<$ on its child edges that is existentially MSOL-definable with 7 parameters.*

Proof. Let $G = (V, E)$ be a Halin graph with an orientation on its edges as shown in Proposition 4.2, E_T its edges of the tree $T = (V, F)$ of G, E_C the edges of the outer cycle and r the root of the tree E_T. Now, consider an inner vertex $v \in V$ (a non-leaf vertex w.r.t. the tree) and two child edges e and f of v (with $e \neq f$). Every edge of

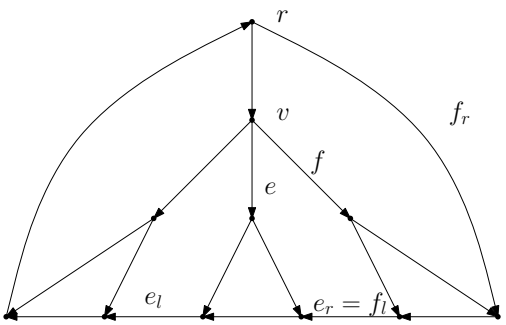


Figure 4.1: Example of a Halin graph with edge orientation.

a Halin graph is contained in exactly two fundamental cycles (w.r.t. T). Suppose we have an ordering on the child edges of v and f is the right neighbor of e. We denote the edges in E_C, whose fundamental cycles contain e and f by e_ℓ, e_r, f_ℓ and f_r, such that e_ℓ and f_ℓ (e_r and f_r) are contained in the left (right) fundamental cycles of e and f, respectively. (See Figure 4.1 for an example.)

Now consider directed paths in E_C from r to the tail vertices of the above mentioned edges. If f is on the right-hand side of e, then the path from r to the tail of f_r is always the shortest of the four. The parameters are the same ones as in Proposition 4.2. The MSOL-predicates given in Appendix A.2 define such an ordering $\mathrm{nb}_<(e, f)$. □

4.2 MSOL-Definable Tree Decompositions

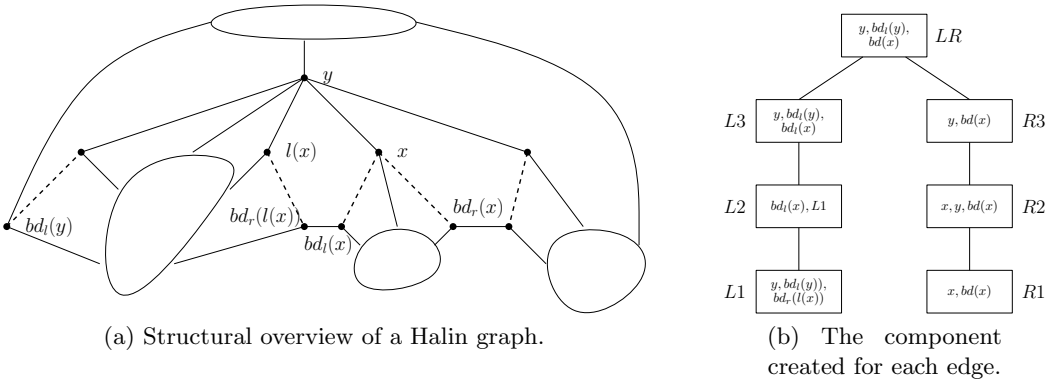
In this section we will describe how to construct a width-3 tree decomposition of a Halin graph, which is definable in monadic second order logic.

First we introduce the notion of *left* and *right boundary* vertices of a Halin graph with an edge orientation and ordering as described in the previous section.

Definition 4.4 (Left and Right Boundary Vertex)**.** Let $G = (V, E)$ be a Halin graph and $v \in V$. A vertex $w \in V$ is called the *left boundary vertex* of v, denoted by $w = bd_l(v)$, if it lies on the cycle of G and there exists a (possibly empty) path E_P from v to $bd_l(v)$ in E_T, such that the tail vertex of each edge in E_P is the leftmost child of its parent. Similarly, we define a *right boundary vertex* $bd_r(v)$. The *boundary* of a vertex v is the set containing both its left and right boundary vertex, denoted as $bd(v)$.

Note that for all cycle vertices $v \in V_C$, we have $v = bd_l(v) = bd_r(v)$. We now state the main result of this section.

Lemma 4.5. *Halin graphs admit width-3 existentially MSOL-definable tree decompositions with 7 parameters.*



(a) Structural overview of a Halin graph.

(b) The component created for each edge.

Figure 4.2: Constructing a component of a tree decomposition for an edge of a Halin graph.

Proof. Let $G = (V, E)$ be a Halin graph and suppose we have an orientation and ordering on its edges as described in Section 4.1. That is, we have a partition (E_C, E_T) of E such that E_C forms the (directed) outer cycle and E_T the (directed) tree of G and there is an ordering on edges with the same head vertex in E_T.

For each edge $e \in E_T$ we construct a component in the tree decomposition that covers the edge itself and one edge on the outer cycle. A component for an edge $e = \{x, y\}$, where y is the parent of x in E_T covers the edges $\{x, y\}$ and the edge $\{bd_r(l(x)), bd_l(x)\}$ on E_C, whose fundamental cycle both contains $\{x, y\}$ and $\{l(x), y\}$ (see Figure 4.2a for an illustration). For the former we create a branch of bags of types $R1, R2$ and $R3$ and for the latter bags of types $L1, L2$ and $L3$, joined by a bag of type LR, containing the following vertices.

R1. This bag contains the vertex x and its boundary vertices $bd(x)$.

R2. This bag contains the vertices x and y and the vertices $bd(x)$.

R3. This bag forgets the vertex x and thus contains y and $bd(x)$.

L1. This bag contains the vertices $y, bd_l(y)$ and $bd_r(l(x))$.

L2. This bag introduces the vertex $bd_l(x)$ to all vertices in the bag $L1$.

L3. This bag forgets the vertex $bd_r(l(x))$ and thus contains $y, bd_l(y)$ and $bd_l(x)$.

LR. This bag contains the union of $L3$ and $R3$, and hence contains the vertices $y, bd_l(y)$ and $bd(x)$.

Figure 4.2b illustrates the structure of the component described above.

To continue the construction, we note that removing $bd_r(x)$ from the bag of type LR results in a bag of type $L1$ for the right neighbor edge, if such an edge exists. If x is the rightmost child of y, then removing $bd_r(x)$ results in a bag of type $R1$ for the edge between y and its parent in E_T. This way we can glue together components of edges using the orientation and ordering of the edge set of the graph. Note that if x is the leftmost child of y, then it is sufficient to only create bags of types $R1, R2$ and $R3$, since we do not have to cover an edge on the outer cycle.

Once we reach the root (i.e. y is the root vertex of the graph), we only create the bags of type $R1$ and $R2$ and our construction is complete.

One can verify that this construction yields a tree decomposition of G and since the maximum number of vertices in one bag is four, its width is three.

To show that these tree decompositions are MSOL-definable, we note that we can define each bag type in MSOL in a straightforward way, once we defined a predicate for boundary vertices. The predicate $\text{Parent}(X_p, X_c)$ requires that there are no two bags in the tree decomposition that contain the same vertex set and so we contract all edges between bags with the same vertex set.

We observe that our predicates only use parameters introduced in the previous section (and hence, their number is seven). The MSOL-predicates given in Appendix A.3 complete the proof. □

From the construction given in this proof, we can immediately derive a consequence that will be useful in the proof of Section 4.3.

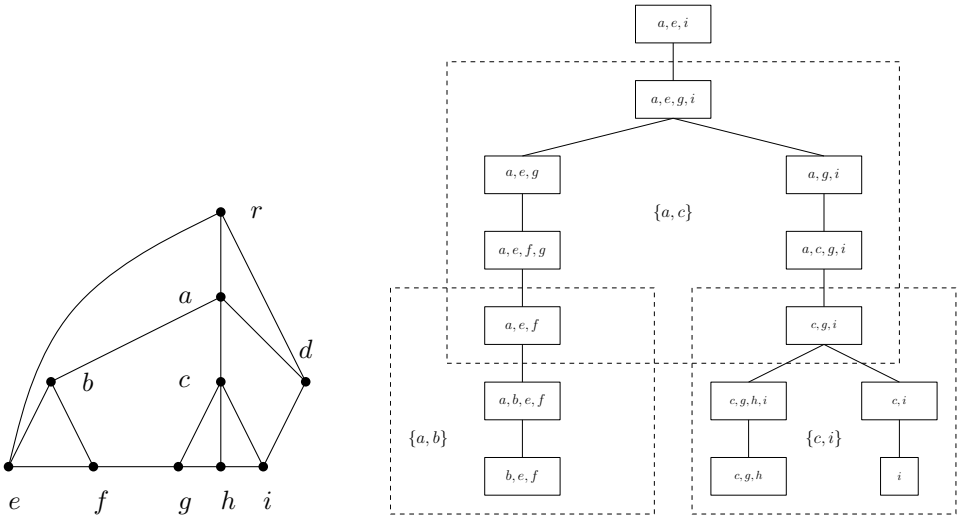
Corollary 4.6. *Halin graphs admit binary width-3 existentially MSOL-definable tree decompositions with seven parameters such that all of their leaf bags have size one.*

Proof. It is easy to see by the construction given in the proof of Lemma 4.5 that this tree decomposition is binary. All leaf bags are of type $R1$ and are associated with edges whose tail vertex x is a vertex on the outer cycle. Hence, $x = bd_l(x) = bd_r(x)$ and our claim follows. □

We will illustrate the construction of a tree decomposition given in the proof of Lemma 4.5 with the following example.

Example 4.7. Consider the graph depicted in Figure 4.3a. We are going to show how to create the component of its tree decomposition corresponding to the edges $\{a, b\}$, $\{a, c\}$ and $\{c, i\}$.

- $\{a, b\}$: Since the vertex b does not have a left sibling, we only create bags $R1$, $R2$ and $R3$. Note that $LR = R3$, since $LR = L3 \cup R3$, and we do not have a bag of type $L3$.
- $\{c, i\}$: Since i is a leaf vertex we have that $bd_l(i) = bd_r(i) = i$ and so the right path starts with a bag $\{i\}$. For the same reason we have that the bags $R2$ and $R3$ are equal and we contract the edge. For the left path this has the effect that $L3$ and LR are equal, so the edge between them gets contracted as well.



(a) An example Halin graph.

(b) The component of the tree decomposition corresponding to the denoted edges.

Figure 4.3: An example subtree of a tree decomposition of a Halin graph.

- $\{a, c\}$: This component can be constructed in a straightforward manner. The bag $L1$ is the parent of the bag LR w.r.t. $\{a, b\}$ and $R1$ is the parent of LR w.r.t. $\{c, i\}$. Since in both cases the vertex sets are equal, we also contract these edges.

Figure 4.3b shows the resulting part of the tree decomposition.

4.3 Finite Index Implies MSOL-Definability

In this section we complete the proof of our first main result, stated below. We will also use ideas that we give here first for extending our results to other graph classes, see Section 5.

Lemma 4.8. *Finite index implies MSOL-definability for Halin graphs.*

Proof. By Lemma 4.5 we know that Halin graphs admit existentially MSOL-definable width-3 tree decompositions. Hence, what is left to show is that we can define the equivalence class membership of terminal subgraphs w.r.t. its bags in monadic second order logic.

We know that the graph property P has finite index, so in the following we will denote the equivalence classes of $\sim_P$ by $C_1, \ldots, C_r$. By Lemmas 3.2 and 3.7 we know that we can derive the equivalence class of a terminal subgraph w.r.t. a node by the equivalence class(es) of terminal subgraphs w.r.t. its descendant nodes in the tree decomposition. Hence, we can conclude that the following two functions exist, also taking into account that our tree decomposition is binary (Corollary 4.6).

Proposition 4.9. *There exist two functions* $f_I : \mathbb{N}_{|r} \times \mathcal{P}_4(V) \to \mathbb{N}_{|r}$ *and* $f_J : \mathcal{P}_2^{\mathcal{M}}(\mathbb{N}_{|r}) \times \mathcal{P}_4(V) \to \mathbb{N}_{|r}$*, such that:*

(i) If X *is an intermediate bag in a tree decomposition with child bag* X_c *and* $[X_c]^+ \in C_i$*, then* $[X]^+ \in C_{f_I(i,X)}$*.*

(ii) If X *is a branch bag with child bags* X_1 *and* X_2*,* $[X_1]^+ \in C_i$ *and* $[X_2]^+ \in C_j$*, then* $[X]^+ \in C_{f(\{i,j\},X)}$*.*

Roughly speaking, these functions can be seen as a representation of the transition function of an automaton that we are given in the original formulation of the conjecture (cf. Theorem 2.18).

Next, we mimic the proof of Büchi's famous classic result for words over an alphabet [8], as shown in [29, Theorem 3.1]. For each equivalence class i we define sets $C^E_{i,\sigma} \subseteq E$ for each type σ (see the proof of Lemma 4.5) and equivalence class i. An edge e is contained in set $C^E_{i,\sigma}$, if and only if the terminal subgraph rooted at a bag of type σ w.r.t. the edge e is in equivalence class i.

Our MSOL-predicate ϕ consists of three parts. First, we identify the equivalence classes corresponding to leaf nodes of the tree decomposition, and we will denote this predicate as ϕ_{Leaf}. This is rather trivial, since we know that all leaf bags contain exactly one vertex (Corollary 4.6) and there is one unique equivalence class to which they all belong, in the following denoted by C_{Leaf}. Note that these bags are always of type $R1$.

Second, we derive the equivalence class membership for terminal subgraphs using Proposition 4.9, assuming we already determined the equivalence class to which the terminal subgraphs w.r.t. its descendants belong. We denote this predicate by ϕ_{TSG}.

Lastly, we check if the graph corresponding to the terminal subgraph of the root bag of the tree decomposition is contained in a P-equivalence class, which we denote by ϕ_{Root}. We know that we can identify these equivalence classes by (the discussion given after) Theorem 2.18 and will denote them by $C_{A_1}, \ldots, C_{A_p}$.

Our MSOL-predicate then combines to:

$$\phi = \phi_{Leaf} \wedge \phi_{TSG} \wedge \phi_{Root} \tag{4.1}$$

Note that ϕ as stated in Equation 4.1 is not parameter-free (as we use an existentially defined tree decomposition). By Lemma 2.23 we know that we can turn ϕ into a sentence Φ given an evaluation Halin graph. This observation together with the details given in Appendix A.4 complete the proof. □

Combining Lemma 4.8 with Theorem 2.18 and [9], we obtain the following.

Theorem 4.10. *MSOL-definability equals recognizability for Halin graphs.*

5

Extensions

The methods we used in the proofs of Chapter 4 can be generalized and applied to a number of other graph classes, some of which we are going to discuss in this section. The main results are presented in Sections 5.1 and 5.2. In the former we show that finite index implies MSOL-definability for any graph class that admits either a bounded degree or an ordered MSOL-definable tree decomposition and in the latter we generalize this result to unordered tree decompositions of unbounded degree using the counting predicate of CMSOL.[(i)] We demonstrate the use of the results of Section 5.1 in Sections 5.3 and 5.4, where we construct ordered or bounded degree MSOL-definable tree decompositions for a subclass of k-outerplanar graphs and for graphs constructed by some bounded size feedback vertex or edge sets, respectively.

5.1 MSOL-Definable Tree Decompositions

We will now turn to generalizing the proof for Halin graphs to any graph class that admits existentially MSOL-definable width-k tree decompositions that are either ordered or have bounded degree. This result will serve as a useful tool to prove Courcelle's Conjecture for a number of graph classes, since it will follow immediately from the construction of such MSOL-definable tree decompositions.

The proof works analogously to the proof of Lemma 4.8 so we will focus on pointing out the differences. We use the same notation.

Lemma 5.1. *Finite index implies MSOL-definability for each graph class that admits existentially MSOL-definable ordered width-k tree decompositions with a constant number of parameters.*

[(i)] For similar findings from the perspective of tree automata see the discussion on page 575 in [16].

Proof. It is easy to see that the predicate ϕ_{Root} can be defined in the same way as in the proof of Lemma 4.8, only adding a short case analysis, since we do not necessarily know of which type the root bag is. Since leaf bags might not always have size one, we apply a small change to the tree decomposition. Assume that its width is k and that we have a $(k+1)$-coloring on the vertices of the graph, such that each vertex in a bag has a different color. Then, for each leaf bag of size greater than one, we add one child bag containing only the vertex with the lowest numbered color. This bag will be identified by a newly introduced type and associated with the same vertex/edge as its parent. We modify the Bag- and Parent-predicates accordingly and can define ϕ_{Leaf} in the same way as in Lemma 4.8, again including a case analysis as for the ϕ_{Root}-predicate.

Hence, in the following we only need to show how to define ϕ_{TSG} to prove the claim. Again we denote the equivalence classes of $\sim_P$ by $C_1, \ldots, C_r$. We can use the function f_I defined in Proposition 4.9 (replacing $\mathcal{P}_4(V)$ by $\mathcal{P}_{k+1}(V)$) to describe the relations between the equivalence classes for intermediate nodes. We need another function to handle partial terminal subgraphs w.r.t. a branch node, whose existence is guaranteed by Lemma 3.2.

Proposition 5.2. *There exists a function $f_J : \mathbb{N}_{|r} \times \mathbb{N}_{|r} \to \mathbb{N}_{|r}$, such that the following holds. If X is a branch bag with child bag Y, $[X]^+_Y \in C_i$ and $[Y]^+ \in C_j$, then:*

(i) If Y is the rightmost child of X, then $[X]^+ \in C_{f_J(i,j)}$.

(ii) Otherwise $[X]^+_{r(Y)} \in C_{f_J(i,j)}$, where $nb_{\prec}(Y, r(Y))$.

In the following, let $\tau \in \{\tau_1, \ldots, \tau_t\}$ and $\sigma \in \{\sigma_1, \ldots, \sigma_s\}$. We define a number of sets, each one associated with an equivalence class i, containing either vertices or edges in the graph (as indicated by their upper indices), $C^V_{i,\tau}$ and $C^E_{i,\sigma}$. If a vertex v is contained in the set $C^V_{i,\tau}$ this means that the terminal subgraph rooted at the bag for vertex v of type τ is in equivalence class i. $C^E_{i,\sigma}$ is the edge set analogous to $C^V_{i,\tau}$. These sets can be used to define the equivalence class membership of terminal subgraphs rooted at intermediate nodes.

Now let X be a bag in the tree decomposition with child Y, such that the node containing X is an intermediate node. We have to distinguish four cases when deriving the membership of a vertex/an edge in the respective sets, which are:

(I) Both X and Y correspond to a vertex.

(II) Both X and Y correspond to an edge.

(III) X corresponds to a vertex and Y to an edge.

(IV) X corresponds to an edge and Y to a vertex.

The predicates defining these cases for intermediate nodes are given in Appendix A.5.1.

When considering a branch node and the partial terminal subgraphs associated with it, we have to analyze at most eight such cases. We first turn to the definition of sets representing the equivalence class membership of a partial terminal subgraph rooted at a branch bag w.r.t. one of its children. Suppose that a bag X is of type τ for a vertex v and one of its child bags Y is of type τ' for the vertex v'. Let $C_{i,\tau}^{V|P}$ and $C_{i,\tau'}^{V|C}$ be sets of vertices. We express that the partial terminal subgraph rooted at the bag of type τ for vertex v w.r.t. the bag of type τ' for vertex v' is in equivalence class i by having $v \in C_{i,\tau}^{V|P}$ and $v' \in C_{i,\tau'}^{V|C}$. We define edge sets $C_{i,\sigma}^{E|P}$ and $C_{i,\sigma}^{E|C}$ with the same interpretation. Also for branch nodes we do a case analysis as indicated above and the predicates in Appendix A.5.2 complete the proof. □

If we are given an MSOL-definable tree decomposition that does not have an ordering on the children of branch nodes, but instead we know that each branch node has a constant number of children, we can prove a similar result.

Lemma 5.3. *Finite index implies MSOL-definability for each graph class that admits existentially MSOL-definable bounded degree width-k tree decompositions with a constant number of parameters.*

Proof. Since this proof works almost exactly as the proof of Lemma 5.1, we only state the differences. Let $c+1$ denote the maximum degree of a (branch) node in the tree decomposition and again we refer to the equivalence classes of $\sim_P$ as $C_1, \ldots, C_r$. Using Lemma 3.7 we know that the following holds (generalizing Proposition 4.9(ii)).

Proposition 5.4. *There exists a function $f_J : \mathcal{P}_c^{\mathcal{M}}(\mathbb{N}_{|r}) \times \mathcal{P}_{k+1}(V) \to \mathbb{N}_{|r}$, such that if X is a branch bag in a tree decomposition with child bags $X_1, \ldots, X_k$ (where $2 \leq k \leq c$), and each terminal subgraph $[X_i]^+$ is in equivalence class C_{c_i}, then the terminal subgraph $[X]^+$ is in equivalence class $C_{f_J(\{c_1,\ldots,c_k\},X)}$.*

Again, to define our predicate we use vertex sets $C_{i,\tau}^{V}$ to represent equivalence class membership of a terminal subgraph rooted at a vertex bag of type τ and edge sets $C_{i,\sigma}^{E}$ for edge bags of type σ (and equivalence class i). We show how to define a predicate for branch bags in such tree decompositions in Appendix A.5.3 and our claim follows. □

Combining Lemmas 5.1 and 5.3 with Theorem 2.18 and [9], we obtain the following.

Theorem 5.5. *MSOL-definability equals recognizability for graph classes that admit existentially MSOL-definable ordered or bounded degree width-k tree decompositions with a constant number of parameters.*

5.2 Unordered Nodes of Unbounded Degree

In the previous section we have shown that for each graph class, whose members admit MSOL-definable tree decompositions, which are either ordered or have bounded degree, finite index implies MSOL-definability. We will now prove that using the counting predicate of CMSOL, one can determine the equivalence class membership of an unordered branch node of unbounded degree as well. Hence, the main result of this section is the following.

Lemma 5.6. *Finite index implies CMSOL-definability for each graph class that admits existentially MSOL-definable width-k tree decompositions with a constant number of parameters.*

Proof. By the proof of Lemma 5.1 we know that the only thing we need to show is how to handle the case when a branch node in the tree decomposition has an unbounded number of children. The outline of the proof is as follows. We group the children of a branch node X of unbounded degree by the equivalence class, their terminal subgraphs are contained in. Then, we show that we can derive the equivalence class membership of the terminal graph consisting of the union of X and the graphs in a group, with terminal set X, by a constant-length function. Using this result we can determine the equivalence class of the terminal subgraph of X. We conclude the proof by showing that this construction is CMSOL-definable.

Definition 5.7 (Group of an Equivalence Class, Partial Terminal Group Subgraph)**.** Let X be a branch bag with an unbounded number of children.

(i) A set of bags is called the *group* i w.r.t. an equivalence class C_i, denoted by $\mathcal{G}_i^X$, if it contains all children of X, whose terminal subgraphs are in equivalence class C_i.

(ii) The terminal graph consisting of X and the union of all terminal subgraphs of a group i with terminal set X is called the *partial terminal group subgraph* of X w.r.t. C_i, denoted by $[X]^+_{\mathcal{G}_i}$.

Suppose X_G is such a branch bag and $X_{H_1}, \ldots$ are its children. Consider an equivalence class C_i and all nodes $X_{H_1}, \ldots, X_{H_\ell}$ with $[X_{H_i}]^+ \in C_i$ (hence, they form the group $\mathcal{G}_i^{X_G}$). By Proposition 4.9 we know that the following holds (i.e. we view each bag as the only child of X_G).

$$\underbrace{[X_{H_1}]^+ \oplus_T X_G}_{H'_1} \in C_{f_I(i,X_G)}, \ldots, \underbrace{[X_{H_\ell}]^+ \oplus_T X_G}_{H'_\ell} \in C_{f_I(i,X_G)}$$

By the arguments given in Section 3, we know that the following function exists.

Proposition 5.8. *Let X be a branch bag in a tree decomposition as described above. Let G and H be two terminal graphs with terminal set X and their vertex and edge sets being the union of some terminal subgraphs w.r.t. child bags of X. Then there exists a function $f_J^{\mathcal{P}_{k+1}(V)} : \mathbb{N}_{|r} \times \mathbb{N}_{|r} \to \mathbb{N}_{|r}$, such that if G is in equivalence class i and H is in equivalence class j, then $G \oplus_{\triangleright} H$ is in equivalence class $f_J^X(i,j)$. We might drop the upper index X if it is clear from the context.*

Let H_1', H_2' and H_3' be terminal graphs as shown above and $j = f_I(i, X_G)$. By Proposition 5.8 we know that $H_1' \oplus_{\triangleright} H_2'$ is in equivalence class $f_J^{X_G}(j,j)$ and subsequently, $H_1' \oplus_{\triangleright} H_2' \oplus_{\triangleright} H_3'$ is in equivalence class $f_J^{X_G}(f_J^{X_G}(j,j),j)$. We define a recursive function as follows.

Definition 5.9. Let X be a branch bag in a tree decomposition and f_J^X as above. Then, for an equivalence class j, we let

$$f_J^1(j) = f_J^X(j,j), \text{ and}$$
$$f_J^n(j) = f_J^X(f_J^{n-1}(j), j), \text{ where } n \in \mathbb{N}, n > 1.$$

It is easy to see that for any fixed j the sequence $f_J^1(j), \ldots$ becomes periodic at some point, since the number of values, which $f_J(i,j)$ can have, is bounded by the constant r and for any i, i' with $i = i'$, $f_J(i,j) = f_J(i',j)$. However, the length of this period depends on the property P and the equivalence class under consideration. We bound the length of the period (somewhat pessimistically) by $r!$, since $r!$ is the product of all possible period lengths over r values.

Proposition 5.10. *For any $j = 1, \ldots, r$ and $n \in \mathbb{N}$ we have*

$$f_J^n(j) = f_J^{n \bmod r!}(j).$$

By Proposition 5.10, we can restrict ourselves in the following to constant values c, where $c = n \bmod r!$, when evaluating f_J^n. Hence, we know that for each group $\mathcal{G}_i^X$, we can determine the equivalence class of each partial terminal group subgraph $[X]_{\mathcal{G}_i}^+$ using a function of constant length.

Proposition 5.11. *Let X be a branch bag with an unbounded number of children. There exists a function $f_{\mathcal{G}} : (\epsilon \cup \mathbb{N}_{|r})^r \to \mathbb{N}_{|r}$, such that if $[X]_{\mathcal{G}_1}^+ \in C_{i_1}, \ldots, [X]_{\mathcal{G}_r}^+ \in C_{i_r}$, then $[X]^+ \in C_{f_{\mathcal{G}}(i_1,\ldots,i_r)}$, where the argument ϵ denotes that the corresponding group is empty.*

Proof. We split the edges of the node containing X in the following way. For each (nonempty) group i we create one new bag X_i with $X_i = X$ and make it adjacent to X. Then, we delete the edges between X and all bags in group i and instead make them adjacent to X_i. Clearly, $[X]_{\mathcal{G}_i}^+ = [X_i]^+$. Since now, X has bounded degree of at most r we can use Lemma 3.7 to conclude the proof. □

We now turn to defining the equivalence class membership of partial terminal group subgraphs in CMSOL. For each vertex bag type τ and pair of indices $i = 1, \ldots, r$ and $j = \epsilon, 1, \ldots, r$ (where ϵ represents the case then group i is empty), we define a set $C_{j,\tau}^{\mathcal{G}_i|V}$, such that the partial terminal group subgraph of X w.r.t. the equivalence class C_i is in equivalence class C_j, if and only if $v \in C_{j,\tau}^{\mathcal{G}_i|V}$. We define edge sets $C_{j,\sigma}^{\mathcal{G}_i|E}$ with the analogous meaning. Note that we need the counting predicates of CMSOL for determining the equivalence class membership of partial terminal group subgraphs, see Proposition 5.10. The predicates defining these (and the equivalence class membership of an unordered node of unbounded degree) are given in Appendix A.5.4 and complete the proof of Lemma 5.6. □

Combining Lemma 5.6 with Theorem 2.18 and [9] then yields the following.

Theorem 5.12. *CMSOL-definability equals recognizability for all graph classes that admit existentially MSOL-definable width-k tree decompositions with a constant number of parameters.*

5.3 k-Cycle Trees

In this section we consider graph class which can be seen as a slight generalization of Halin graphs. Recall the results of Sections 4.1 and 4.2.

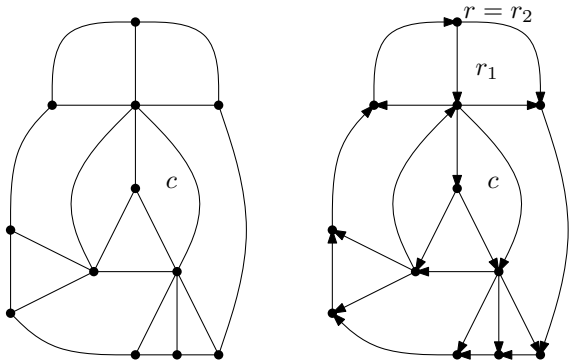
Definition 5.13 (k-Cycle Tree)**.** A graph G is called *cycle tree*, if it can be obtained by a planar embedding of a tree with one distinguished vertex $c \in V$, called the *central* vertex, such that all vertices of distance d from c are connected by a cycle and the embedding stays planar. If each vertex (except for c) is contained in one cycle, the number of which is k, then G is called a k-cycle tree. We will refer to the cycle of distance d from c as the cycle C_d.

Figure 5.1a shows an example of a 2-cycle tree. We easily observe the following.

Proposition 5.14. *Each k-cycle tree is $(k+1)$-outerplanar.*

Since k-outerplanar graphs have treewidth at most $3k - 1$ [4, Theorem 83], we know by 5.14 that we can apply all results of treewidth k graphs to k-cycle trees as well. To construct an MSOL-definable tree decomposition for a k-cycle tree, we need the notion of the i-th left and right boundary of a vertex, referring to vertices on the i-th cycle of the graph.

Definition 5.15 (i-th boundary vertex)**.** Let $G = (V, E)$ be a k-cycle tree and $v \in V$. We say that w is the *i-th left boundary vertex* of v, denoted by $w = bd_i^l(v)$, if w lies on C_i and there exists a path E_P^l from v to w, only using edges of the tree of the graph,



(a) G without edge orientation

(b) G with edge orientation

Figure 5.1: An example 2-cycle tree G with central vertex c.

such that no other path from v to any vertex on C_i exists that uses an edge that lies on the left of one of the edges in E_P^l. Similarly, we define the *i-th right boundary vertex* $bd_i^r(v)$.

Now we are ready to prove the main result of this section.

Lemma 5.16. *k-Cycle trees admit existentially MSOL-definable binary width-$4k$ tree decompositions with $4k + 3$ parameters.*

Proof. We can show this in almost exactly the same way as for Halin graphs (Lemma 4.5), so we will focus on pointing out the differences. Again, at first we define an edge orientation on k-cycle trees. Instead of partitioning the edge set into one directed tree and one directed cycle we now have one directed tree E_T and k directed cycles, such that E_{C_i} denotes the cycle of distance i from the central vertex c.

The root of the tree is a vertex incident to the outermost cycle and for each cycle C_i we have one incident root vertex r_i, which will be used to define the neighbor ordering of edges with the same head vertex. For a cycle C_i this will be a vertex of distance $k - i$ from the root vertex of the tree. One can verify that turning the tree and cycles of G into their directed equivalents is MSOL-definable by Lemma 4.1, Proposition 5.14, and the predicates given in Appendix A.6. For an illustration of the edge orientation see Figure 5.1b.

Using this orientation one can define a predicate $\mathrm{nb}^i_<(e, f)$ for ordering all edges with the same parent, which then can be utilized to define i-th boundary vertices.

As in the proof of Lemma 4.5, we construct a component in the tree decomposition for each edge $e \in E_T$. The definition of the bag types is somewhat different, since now we have to take into account at most k cycle edges per component instead of a single one. Given an edge $e = \{x, y\} \in E_T$ such that y is the parent of x and y lies on cycle

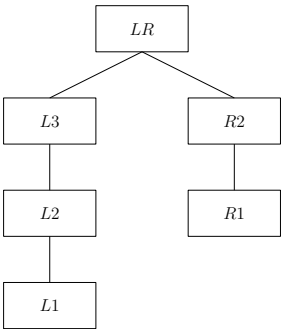


Figure 5.2: Bag types and edges for a component in the tree decompositions of a k-cycle tree.

C_i, we have the following types of bags, with edges between them as shown in Figure 5.2. (Note that if in the following we refer to boundary vertices, we always mean the boundary vertices on higher numbered cycles.)

R1. This bag contains the vertex x and all its left and right boundaries.

R2. This bag contains all vertices in the bag $R1$ plus the vertex y.

L1. This bag contains the vertex y, all its left boundary vertices and the right boundary vertices of y in the forest consisting of E_T without the edge e and its right neighbors.

L2. This bag contains all vertices of the bag $L1$ plus the left boundary vertices of x (including x itself, if $x \neq c$).

L3. This bag contains the vertices of the bag $L2$ minus the right boundary vertices z of y without e and its right neighbors, such that z has a matching left boundary vertex. That is, there is an edge between said boundary vertices and thus the vertex z can be forgotten.

LR. This bag contains the union of the bags $L3$ and $R2$.

One can verify that this construction yields a tree decomposition for k-cycle trees. The largest of its bags is of type LR, which might contain four boundary sets, each of which has size at most k, plus the vertices x and y. Since we have only one vertex, which is no boundary vertex (the central vertex c), we can conclude that the size of this bag is at most $4k + 1$ and hence this tree decomposition has width $4k$.

We now count the number of parameters. For the edge orientation we need $3k + 1$ parameters (by Lemma 4.1, Proposition 5.14 and [4, Theorem 83]), we have one edge set for the spanning tree, k edge sets for the cycles, and we can use one vertex set for the roots of the cycles. Hence, the total number of parameters is $4k + 3$. The predicates in Appendix A.6 complete the proof. □

Combining Lemma 5.16 with Theorem 5.5, we can derive the following.

Theorem 5.17. *MSOL-definability equals recognizability for k-cycle trees.*

5.4 Feedback Edge and Vertex Sets

In this section we consider graphs that can be obtained by the composition of a graph that admits an existentially MSOL-definable (ordered) tree decomposition and some feedback edge or vertex sets, defined below.

Definition 5.18. Let $G = (V, E)$ be a graph. An edge set $E' \subseteq E$ is called *feedback edge set*, if $G' = (V, E \setminus E')$ is acyclic. Analogously, a vertex set V' is called *feedback vertex set*, if the graph $G' = (V \setminus V', E \setminus E')$ is acyclic, where E' denotes the set of incident edges of V' in E.

Theorem 5.19. *Let $G = (V, E)$ be a graph (and $T = (V, F)$ a spanning tree of G), which admits an existentially MSOL-definable (ordered or bounded degree) width-k tree decomposition with p parameters, such that its vertex and edge bag predicates are associated with either (a subset of the) vertices of the graph or (a subset of the) edges in the spanning tree.*

Let l be a constant. A graph G' admits an existentially MSOL-definable (ordered or bounded degree) tree decomposition of width $k + l$ with at most $p + k + 1$ parameters, if one of the following holds.

(i) Let E' denote a set of edges, such that each biconnected component of the graph $T' = (V, F \cup E')$ has a feedback edge set of size at most l, where $G' = (V, E \cup E')$.

(ii) Let V' denote a set of vertices and $E' \subseteq (V \times V') \cup (V' \times V')$ a set of incident edges, such that each biconnected component of the graph $T' = (V \cup V', F \cup E')$ has a feedback vertex set of size at most l, where $G' = (V \cup V', E \cup E')$.

Proof. (i). Let $e = \{v, w\}$ be an edge in E' and note that since G has bounded treewidth k, there exists a $(k + 1)$-coloring on its vertices. Assume wlog. that the coloring set is a set of natural numbers $\{1, \ldots, k + 1\}$ and $\mathrm{col}(v) < \mathrm{col}(w)$. Then we add the vertex v to each bag that is associated with either a vertex or an edge in T that lie on the fundamental cycle of e. The width of the tree decomposition increased by at most l (by Lemmas 6 and 73 in [4]).

(ii). Let v be a vertex in V'. We add v to all bags that correspond to vertices/edges contained in the same biconnected component as v (in T'). The fact that the treewidth increased by at most l follows from [4, Lemmas 6 and 72].

In Appendix A.7 we show how to extend all predicates to include the newly introduced vertices in the bags for both cases. In (i), the number of parameters used might increase by $k + 1$, since we don't know whether we already know a coloring of the graph. □

Note that by our construction, one can apply Theorem 5.19 to both Halin graphs and k-cycle trees.

6

k-Outerplanar Graphs

In this chapter we investigate the class of k-outerplanar graphs (see Definition 2.3). As a warm up, we will show that finite index implies MSOL-definability for bounded degree k-outerplanar graphs (Section 6.1). We will prove the same result for 3-connected k-outerplanar graphs (Sections 6.2 and 6.3). We resolve Courcelle's Conjecture for the general case of k-outerplanar graphs in Section 6.4, showing that finite index implies CMSOL-definability. In our proof we use decompositions of graphs into 3-connected components due to Tutte [30], which have been proven to be MSOL-definable by Courcelle [14].

Bodlaender has shown that every k-outerplanar graph has treewidth at most $3k-1$ [4, Theorem 83], using the following properties of maximal spanning forests of a graph.

Definition 6.1 (Vertex and Edge Remember Number)**.** Let $G = (V, E)$ be a graph with maximal spanning forest $T = (V, F)$. The *vertex remember number* of G (with respect to T), denoted by $vr(G, T)$, is the maximum number over all vertices $v \in V$ of fundamental cycles (in G given T) that use v. Analogously, we define the *edge remember number*, denoted by $er(G, T)$.

In particular, Bodlaender gave a constructive proof that the treewidth of a graph is bounded by at most $\max\{vr(G, T), er(G, T) + 1\}$ [4, Theorem 71] (see also the proof of Theorem 6.2). The idea of the proof is to create a bag for each vertex and edge in the spanning tree, containing the vertex itself (or the two endpoints of the edge, respectively) and one endpoint of each edge, whose fundamental cycle uses the corresponding vertex/edge. The tree structure of the decomposition is inherited by the structure of the spanning tree. He then showed, that in a k-outerplanar graph G one can split the vertices of degree $d > 3$ into a path of $d-2$ vertices of degree three without increasing the outerplanarity index of G (the so-called *vertex expansion step*). In this expanded graph G' one can find a spanning tree of vertex remember number at most $3k-1$ and edge remember number at most $2k$ [4, Lemmas 81 and 82]. Using [4, Theorem 71], this yields

a tree decomposition of width at most $3k-1$ for G' and by simple replacements one finds a tree decomposition for G of the same width. A constructive version of this proof was given by Katsikarelis [23]. The expansion step is the major challenge in defining a tree decomposition of a k-outerplanar graph in monadic second order logic, since we cannot use these newly created vertices as variables. We find an implicit representation of this step in Section 6.2.

However, if G has bounded degree, we do not need to define this step to find a bounded width tree decomposition of G. We will start by investigating this case. In the following, unless stated otherwise, we refer to $G = (V, E)$ as a simple connected k-outerplanar graph.

6.1 Bounded Degree k-Outerplanar Graphs

To prove Courcelle's Conjecture for bounded degree k-outerplanar graphs, we show that the tree decomposition constructed in the above mentioned proof by Bodlaender, whose width is bounded by the vertex and edge remember number of a graph, is existentially MSOL-definable.

Theorem 6.2. *Let $G = (V, E)$ be a graph with a spanning tree $T = (V, F)$ and let $k = \max\{vr(G, T), er(G, T) + 1\}$. G admits*

(i) an existentially MSOL-definable width-k tree decomposition of bounded degree with $k + 3$ parameters, if G has bounded degree.

(ii) an existentially MSOL-definable ordered width-k tree decomposition with a constant number of parameters, if there is an (existentially) MSOL-definable ordering $nb_<(e, f)$ (with a constant number of parameters) over all edges $e, f \in F$ with the same head vertex.

(iii) an existentially MSOL-definable width-k tree decomposition with $k+3$ parameters.

Proof. For (i), (ii) and (iii) we can construct a tree decomposition (T', X) as shown in the proof of Theorem 71 in [4]. That is, we create a tree $T' = (V \cup F, F')$, where $F' = \{\{v, e\} \mid v \in V, e \in F, \exists w \in V : e = \{v, w\}\}$, i.e. we add an extra node between each two adjacent vertices in the spanning tree. The construction of the sets $X_t, t \in V \cup F$ works as follows. For a bag associated with a vertex v in the spanning tree we first add v to X_v, and for a bag associated with an edge e, we add both its endpoints to X_e. Then, for each edge $e \in E \setminus F$, we add one of its endpoints to each bag corresponding to a vertex or edge on the fundamental cycle of e. To make sure that our method of choosing one endpoint of an edge is MSOL-definable, we use the same argument as in

the proof of Theorem 5.19(i). That is, we assume the existence of a vertex coloring in the graph and pick the vertex with the lower numbered color.

One can verify that (T', X) is a tree decomposition of G and we have for all vertex bags X_v that $|X_v| \leq 1 + vr(G, T)$ and for all edge bags X_e that $|X_e| \leq 2 + er(G, T)$ and thus the claimed width of (T', X) follows.

Now we show that finding a spanning tree such that its vertex and edge remember number are bounded by a constant, say κ, is MSOL-definable, if it exists. We can simply do this by guessing an edge set $E_T \subseteq E$ and checking whether E_T is the edge set of a spanning tree in G with the claimed bound on the resulting vertex and edge remember numbers. Since κ is constant, this can be done in a straightforward way, see Appendix A.8.

For defining the Bag- and Parent-predicates, we assume wlog. that we have a root and an MSOL-definable orientation on the edges in the spanning tree,[(i)] so we can directly define such predicates, see Appendix A.8.

For (i) one easily sees that (T', X) has bounded degree, since the degree of any node corresponding to a vertex $v \in V$ in the tree decomposition is equal to the degree of v in G. Nodes containing edge bags are always intermediate nodes. As parameters we have to guess the set of edges of the spanning tree, furthermore we need $k + 1$ color classes (to choose one vertex per edge for a fundamental cycle) and one more edge set for the edge orientation of the spanning tree.

We observe that (ii) holds, since we can define an orientation $\mathrm{nb}_<(X_a, X_b)$ for the children of each vertex bag by using the ordering of its corresponding edges. The number of parameters is the same as in (i), plus the number of parameters used for the ordering $\mathrm{nb}_<$, which is required to be constant.

(iii) follows from observations made in the proof of (i).

The predicates defined in Appendix A.8 complete the proof. □

In his proof for the treewidth of k-outerplanar graphs being $3k-1$, Bodlaender used the following lemma.

Lemma 6.3 (Lemma 81 in [4])**.** *Let $G = (V, E)$ be a k-outerplanar graph with maximum degree 3. Then there exists a maximal spanning forest $T = (V, F)$ with $er(G, T) \leq 2k$ and $vr(G, T) \leq 3k - 1$.*

By the ideas used in its proof, one immediately has the following consequence.

[(i)] This clearly holds by Lemma 4.1, since trees have treewidth 1.

Corollary 6.4. *Let $G = (V, E)$ be a k-outerplanar graph with maximum degree Δ. Then there exists a maximal spanning forest $T = (V, F)$ with $er(G, T) \leq 2k$ and $vr(G, T) \leq \Delta k - 1$.*

We can now prove the main result of this section.

Theorem 6.5. *MSOL-definability equals recognizability for k-outerplanar graphs of bounded degree.*

Proof. Let $G = (V, E)$ be a k-outerplanar graph with maximum degree Δ. By Corollary 6.4, we know that there exists a maximal spanning forest $T = (V, F)$ of G with $er(G, T) \leq 2k$ and $vr(G, T) \leq \Delta k - 1$. By Theorem 6.2(i), we know that G admits an MSOL-definable tree decomposition of bounded degree. If $\Delta < 3$, then the width of this tree decomposition is at most $4k+1$, and if $\Delta \geq 3$, it is at most $\Delta k - 1$, so in both cases the width is bounded by a constant. The rest now follows from Theorem 5.5. □

6.2 An Implicit Representation of the Vertex Expansion Step

As outlined before, the central step in constructing a width-$(3k-1)$ tree decomposition of a k-outerplanar graph G is splitting the vertices of degree $d > 3$ into a path of $d-2$ vertices of degree 3 without increasing the outerplanarity index of the graph G (see the introduction of this chapter). Since we can't mimic this expansion step in MSOL directly, we have to find another characterization of this method, the first step of which is to partition the vertices of a k-outerplanar graph into its *stripping layers*.

Definition 6.6 (Stripping Layer of a k-Outerplanar Graph)**.** Let G be a k-outerplanar graph. Removing the vertices on the outer face of an embedding of G is called the *stripping step*. When applied repeatedly, the set of vertices being removed in the i-th stripping step is called the i-th *stripping layer* of G, where $1 \leq i \leq k$.

Lemma 6.7. *Let $G = (V, E)$ be a k-outerplanar graph. The partition of V into the stripping layers of G is existentially MSOL-definable with k parameters.*

Proof. We first introduce another characterization of stripping layers of k-outerplanar graphs, which we can use later to define our predicates.

Proposition 6.8. *Let $G = (V, E)$ be a k-outerplanar graph. A partition $V_1, \ldots, V_k$ of V represents its stripping layers, if and only if:*

(i) $G[V_i]$ is an outerplanar graph for all $i = 1, \ldots, k$.

(ii) For each vertex $v \in V_i$, all its adjacent vertices are contained in either V_{i-1}, V_i or V_{i+1}.

Proof. ($\Rightarrow$) Since in each step we remove the vertices on the outer face of the graph, it is easy to see that (i) holds. For (ii), suppose not. Wlog. assume that $v \in V_i$ has a neighbor w in V_{i+2}. Before stripping step i, v lies on the outer face. Now, for w to not lie on the outer face after stripping step i, there needs to be a cycle crossing the edge $\{v, w\}$, hence the embedding of G is not planar and we have a contradiction.

($\Leftarrow$) We use induction on k. The case $k = 1$ is trivial. Now assume that $G = (V, E)$ is an ℓ-outerplanar graph with a partition of V into $V_1, \ldots, V_\ell$ such that our claim holds. Let $V_{\ell+1}$ be a set of vertices with neighbors only in $V_{\ell+1}$ and V_ℓ. We denote this edge set by $E_{\ell+1}$. Clearly, placing the vertices in V_ℓ on the outer face results in an $(\ell+1)$-outerplanar embedding of the graph $G' = (V \cup V_{\ell+1}, E \cup E_{\ell+1})$. However, some vertices in V_ℓ might still lie on the outer face. Denote this vertex set by V_ℓ^O. We let $V'_{\ell+1} = V_{\ell+1} \cup V_\ell^O$ and $V'_\ell = V_\ell \setminus V_\ell^O$. Then, the partition $V_1, \ldots, V_{\ell-1}, V'_\ell, V'_{\ell+1}$ satisfies our claim and the result follows (reversing the indices of the sets in the partition). □

It is well known that a graph is outerplanar if it does not contain K_4, the clique of four vertices, and $K_{2,3}$, the complete bipartite graph on two and three vertices, as a minor (cf. [17, p. 112], [28]). Borie et al. showed that the fixed minor relation is MSOL-definable [7, Theorem 4], so in our definition we use the predicates Minor_{K_4} and $\text{Minor}_{K_{2,3}}$ for stating the respective minor containment. The rest can be done in a straightforward way according to Proposition 6.8. The details of the predicates can be found in Appendix A.9, which conclude the proof of Lemma 6.7. □

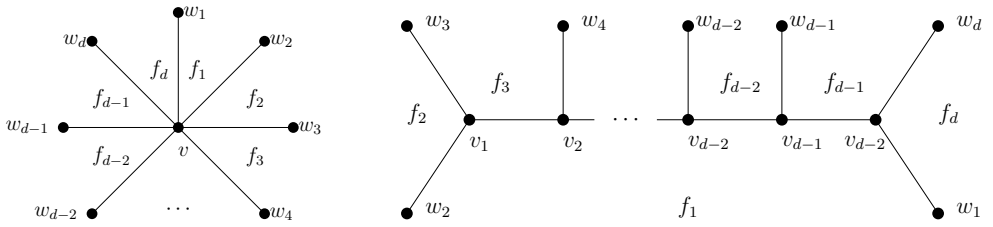
Definition 6.9 (Layer Number)**.** Let $G = (V, E)$ be a planar graph. The *layer number* of a face is defined in the following way. The outer face gets layer number 0. Then, for each other face, we let the layer number be one higher than the minimum layer number of all its adjacent faces.[(ii)]

Proposition 6.10. *Let $G = (V, E)$ be a k-outerplanar graph, $V_1, \ldots, V_k$ its stripping layers and $v \in V_i$. Each face f incident to v has either layer number i or $i - 1$. Furthermore, f has layer number $i - 1$, if the boundary of f contains a vertex w with $w \in V_{i-1}$.*

Proof. We observe that removing all vertices on the outer face makes a face of layer number i become a face of layer number $i - 1$ and our claim follows. □

The expansion step does not preserve facial adjacency, so in order to not increase the outerplanarity index of the graph, one makes sure that all faces are adjacent to a face with

[(ii)] Unless stated otherwise, we call to faces adjacent, if they share an incident vertex.



(a) before expansion (b) after expansion

Figure 6.1: Expanding a vertex v, where f_1 is a layer with lowest layer number.

lowest layer number. We illustrate the expansion step of a vertex in Figure 6.1. Following the ideas of the proofs given in [4, Section 13], we define another type of *remember number* to implicitly represent the expansion step for creating a tree decomposition of a k-outerplanar graph.

Definition 6.11 (Face Remember Number)**.** Let $G = (V, E)$ be a planar graph with a given embedding $\mathcal{E}$ and $T = (V, F)$ a maximal spanning forest of G. The *face remember number* of G w.r.t. T, denoted by $fr(G, T)$ is the maximum number of fundamental cycles C of G given T, such that $bd_E(f) \cap E(C) \neq \emptyset$, where $bd_E(f)$ denotes the boundary edges of a face f, over all faces f in $\mathcal{E}$, excluding the outer face.

For an illustration of face remember numbers, see Figure 6.2. Consider the vertex v_1 in Figure 6.1b and let e be an edge whose fundamental cycle C_e uses v_1 in some spanning tree of G'. We observe that C_e intersects with one of the face boundaries of f_1, f_2 or f_3. Since v_1 is a vertex in the expanded graph, we know that in each tree decomposition based on a spanning tree of G' there will be a bag containing one endpoint of each edge, whose fundamental cycle intersects with the face boundary of f_1, f_2 or f_3. Using this observation, we can also show that one can find a tree decomposition of a planar graph, whose width is bounded by the face remember number of a maximal spanning forest, without explicitly expanding vertices.

Lemma 6.12. *Let $G = (V, E)$ be a planar graph with maximal spanning forest $T = (V, F)$. The treewidth of G is at most* $\max\{er(G, T) + 1, 3 \cdot fr(G, T)\}$.

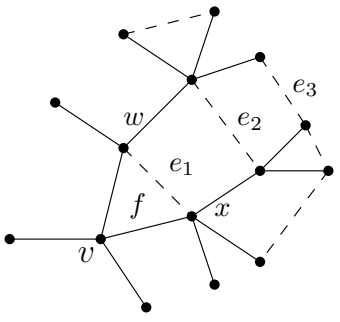


Figure 6.2: A spanning tree of a planar graph with some additional edges (dashed lines). The remember number of the face f, bounded by $bd(f) = \{v, w, x\}$, is 3 in this graph, since the fundamental cycles of the edges e_1, e_2 and e_3 intersect with $bd_E(f)$.

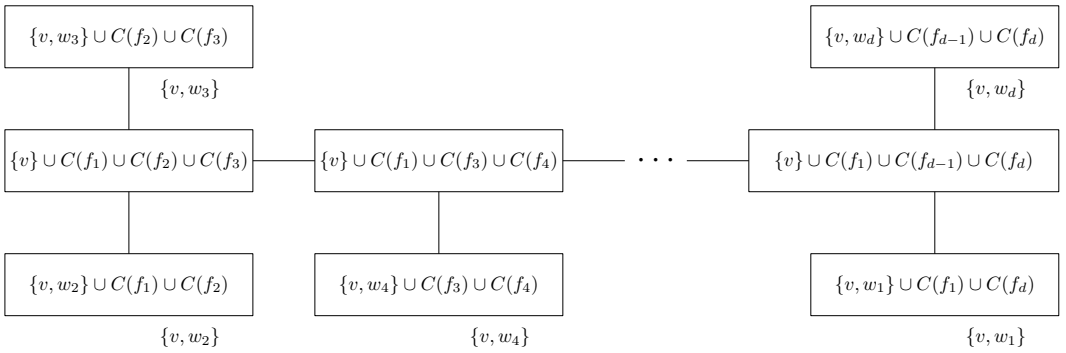


Figure 6.3: A component of a tree decomposition corresponding to a vertex, as used in the proof of Lemma 6.12 (assuming, for explanatory purposes, that all incident edges of v are contained in the maximal spanning forest of the graph).

Proof. Recall the vertex expansion step and see Figure 6.1 for an illustration. In the following, we will construct a tree decomposition (T, X) of the unexpanded graph G, imitating the ideas of the expansion step. That is, for each vertex $v \in V$ we create a path in (T, X) in the following way. First, we add v to each of these bags. Let f_1 denote a face with lowest layer number of all faces incident to v and let all face indices be as depicted in Figure 6.1a.[(iii)] Let $C(f_i)$ denote the set, containing one endpoint of each edge $e \in E \setminus F$, whose fundamental cycle C_e intersects with the edge set of the boundary of the face f_i, i.e. $bd_E(f_i) \cap E(C_e) \neq \emptyset$. Let $\deg(v) = d$. We create bags containing the vertices in $C(f_1) \cup C(f_i) \cup C(f_{i+1})$, where $i = 2, \ldots, d-1$. (For an edge e_i incident to v, f_i and f_{i+1} are its incident faces.) We make two bags adjacent, if they share two sets $C(f_i)$ and $C(f_j)$ and belong to the same vertex. Note that this way we precisely imitate the construction of bags for the artificially created vertices during the expansion step.

Furthermore, for each edge $e \in F$, we create a bag containing both its endpoints and one endpoint of each edge e_{fc}, whose fundamental cycle uses e. We observe that the set $C(f_i) \cup C(f_j)$ contains precisely one vertex for each such edge e_{fc}, where f_i and f_j are the two faces incident to e. We then make this bag adjacent to each bag created in the step before, which corresponds to both $C(f_i)$ and $C(f_j)$ and one more set $C(f')$. For each incident vertex there will always be precisely one such bag and hence, each edge bag will have two neighbors in the tree decomposition (one for each endpoint). For an overview of the constructed component, see Figure 6.3.

One can verify that this construction yields a tree decomposition of G, and since we know that by definition $|C(f)| \leq fr(G, T)$ for all faces f (except the outer face) we know that its width is bounded by $\max\{er(G, T) + 1, 3 \cdot fr(G, T)\}$. □

To apply this result to a k-outerplanar graph G, we show that we can find a maximal spanning forest of G of bounded edge and face remember number.

[(iii)] Note that by by Proposition 6.10, this number will be either i or $i - 1$, if $v \in V_i$.

Lemma 6.13. *Let $G = (V, E)$ be a k-outerplanar graph. There exists a maximal spanning forest $T = (V, F)$ of G with $er(G, T) \leq 2k$ and $fr(G, T) \leq k$.*

Proof. The proof can be done analogously to the proof of Lemma 81 in [4]. □

6.3 3-Connected k-Outerplanar Graphs

We now show that the construction of the tree decomposition given in the previous section is existentially MSOL-definable for 3-connected k-outerplanar graphs. Particularly we will make use of the fact that the face boundaries of a 3-connected planar graph be defined by a predicate in monadic second order logic. We will then define an ordering of all incident edges of a vertex to create a path in the tree decomposition as described in the proof of Lemma 6.12.

A classic result by Whitney states that every 3-connected planar graph has a unique embedding [34] (up to the choice of the outer face). Reconstructing this proof, Diestel has shown that the face boundaries of this embedding can be characterized in strictly combinatorial terms.

Proposition 6.14 (P. 4.2.7 in [17])**.** *The face boundaries in a 3-connected planar graph are precisely its non-separating induced cycles.*

We immediately have the following.

Proposition 6.15. *The face boundaries of a 3-connected planar graph are MSOL-definable.*

Proof. We use Proposition 6.14 and define a predicate, which is true if and only if a vertex set V' is the face boundary of a 3-connected planar graph in the following straightforward way.

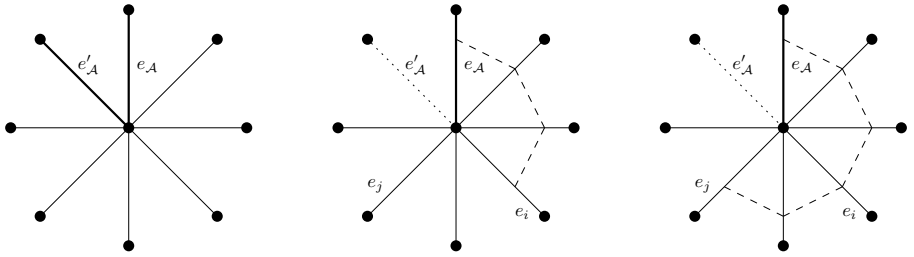
$$\mathrm{FaceBd}_3(V') \Leftrightarrow \mathrm{Cycle}(V', \mathrm{IncE}(V')) \wedge \mathrm{Conn}(V \setminus V', E \setminus \mathrm{IncE}(V'))$$

We can use this predicate to define this notion in terms of edge sets as well.

$$\mathrm{FaceBd}_3(E') \Leftrightarrow \mathrm{FaceBd}_3(\mathrm{IncV}(E'))$$

□

Using these observations, we can define predicates encoding the above mentioned ordering on the incident edges of each vertex. We first need another definition.



(a) A vertex v with the anchor edge $e_{\mathcal{A}}$ and edge $e'_{\mathcal{A}}$.

(b) The path from $e_{\mathcal{A}}$ to e_i, according to face adjacency.

(c) The path from $e_{\mathcal{A}}$ to e_j, according to face adjacency.

Figure 6.4: A vertex v with two edges e_i and e_j, such that $\mathrm{nb}_<(e_i, e_j)$ as described in the proof of Lemma 6.17, defining a clockwise ordering on the incident edges of v. Note that paths in the other direction starting at $e_{\mathcal{A}}$ don't exists, since $e'_{\mathcal{A}}$ can't be included in such a path.

Definition 6.16 (Face-Adjacency of Edges)**.** Let $G = (V, E)$ be a planar graph and $v \in V$. We call two incident edges $e, f \in E$ of v *face-adjacent*, if there is a face-boundary containing both e and f.

Lemma 6.17. *Let $G = (V, E)$ be a 3-connected k-outerplanar graph, $v \in V$ with* $\deg(v) > 3$ *and $e_{\mathcal{A}}$ an incident edge of v, called its* anchor*. There exists an ordering $nb_<(e, f)$, which mimics a clockwise (or counter-clockwise) traversal (in the unique embedding of G) on all incident edges of v, starting at $e_{\mathcal{A}}$, which is existentially MSOL-definable with two parameters $e_{\mathcal{A}}$ and $e'_{\mathcal{A}}$.*

Proof. We first observe an important property of 2-connected planar graphs, which we will use to define the ordering later in the proof.

Proposition 6.18. *Let $G = (V, E)$ be a 2-connected planar graph and $v \in V$. Then, all faces incident to v are pairwise different.*

Proof. Suppose not. Then $\{v\}$ is a separator of G. □

Let $e'_{\mathcal{A}}$ be another incident edge of v, which bounds some face together with $e_{\mathcal{A}}$. (Note that there are exactly two such edges in G, the choice of which decides whether the ordering is clockwise or counter-clockwise.) For any pair of incident edges of v, e_i and e_j, we let $\mathrm{nb}_<(e_i, e_j)$, if and only if we can find sets of edges E_i and E_j with the following properties. Let $\mathrm{Inc}(v)$ denote the set of incident edges of v.

(i) For $\ell = i, j$, the set E_ℓ consists of the edge e_ℓ, $e_{\mathcal{A}}$ and a subset of $\mathrm{Inc}(v) \setminus \{e'_{\mathcal{A}}\}$ and contains precisely all pairs of face-adjacent edges that, according to face-adjacency, form a path from $e_{\mathcal{A}}$ to e_ℓ.

(ii) $E_i \subset E_j$.

For an illustration of the meaning of these edge sets see Figure 6.4. We now turn to defining this ordering in MSOL. By Proposition 6.18, we know that all faces adjacent to v are pairwise different and hence, we can use Proposition 6.15 to define paths in terms of face-adjacency in the unique embedding of G between two incident edges of v. The predicates given in Appendix A.9.1 complete the proof. □

Note that one can lead an alternative proof of Lemma 6.17, using the notion of *rotation systems*, introduced in [15]. Furthermore one can see that the relation $\mathrm{nb}_<(e, f)$ is existentially MSOL-definable for an entire graph G by replacing the parameters in the formulation of Lemma 6.17 with the corresponding edge set equivalents.

Defining the Tree Decomposition

Lemma 6.19. *Let $G = (V, E)$ be a 3-connected k-outerplanar graph. G admits an existentially MSOL-definable tree decomposition of width at most $3k$ and maximum degree 3 with $4k + 4$ parameters.*

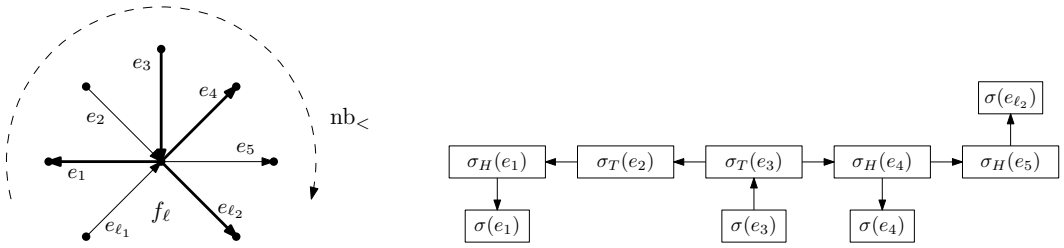
Proof. We mimic the construction given in the proof of Lemma 6.12 and use the same notation. We first prove the definability of the spanning tree, upon which the construction of our tree decomposition is based.

Proposition 6.20. *Let $G = (V, E)$ be a 3-connected k-outerplanar graph. There exists a spanning tree $T = (V, F)$ of G with $er \leq 2k$ and $fr(G, T) \leq k$, which is existentially MSOL-definable with one parameter, the edge set F of T.*

Proof. By Lemma 6.13 we know that such a spanning tree T exists. We can use Proposition 6.15 to define T in MSOL, see Appendix A.9.2. □

We direct the spanning tree T of Proposition 6.20 as shown in Lemma 4.1 to be a rooted tree, using a $3k$-coloring Γ_G of G. Note that two colors would already suffice, but we will later use these color sets to impose an (arbitrary) orientation on the edges in $E \setminus F$ as well.

We now choose the set of anchor and co-anchor edges $E_{\mathcal{A}}$ and $E'_{\mathcal{A}}$, respectively, to fix an ordering on the incident edges of a vertex as shown in Lemma 6.17. For a vertex v, let e_{ℓ_1} and e_{ℓ_2} denote the edges bounding a face f_ℓ with lowest layer number. We then add e_{ℓ_1} to $E_{\mathcal{A}}$ and e_{ℓ_2} to $E'_{\mathcal{A}}$. Hence, we have that $\mathrm{nb}_<(e_{\ell_1}, e)$, for all incident edges e of v.



(a) A vertex with incident (directed) edges. Fat edges are in the spanning tree.

(b) The component in the tree decomposition, where the edge-orientation describes the Parent-relation.

Figure 6.5: A component of a definable tree decomposition as described in the proof of Lemma 6.19, corresponding to a vertex v with a clockwise ordering on its edges, anchored at e_{ℓ_1}, where f_ℓ is a face with lowest layer number of all incident faces of v.

We define three types of bag predicates, all associated with edges. The first type, σ, contains the endpoints of an edge $e \in F$ in the spanning tree of G and one endpoint of each edge, whose fundamental cycle uses e. Note for the following that we can identify an incident face of lowest layer number of each vertex by using Proposition 6.10 (for details see Appendix A.9.2). If there is more than one face with lowest layer number, we choose the one which is closest to the unique incoming edge in v of T (according to $\mathrm{nb}_<$).

We fix an arbitrary orientation on all edges in $E \setminus F$ using the coloring Γ_G together with the empty edge set (see Lemma 4.1). Then we define two more types of bags, σ_H and σ_T for each edge $e_i \in \mathrm{Inc}(v) \setminus \{e_{\ell_1}, e_{\ell_2}\}$ for all $v \in V$. Let $e_i = \{v, w\}$ with orientation from v to w. Then, we create a bag of type σ_H, containing v and one endpoint of each edge in $C(v, f_\ell) \cup C(v, f_{i-1}) \cup C(v, f_i)$,[(iv)] meaning that σ_H is a type associated with the head vertex of an edge. We similarly define a type associated with the tail vertex of an edge, σ_T, which is created in the same way as σ_H, except that it contains the tail vertex instead of the head vertex of e_i (in this case: w).

We now turn to defining the Parent-predicate. For an illustration of any of the below mentioned cases, we refer the reader to Figure 6.5, which gives an example of a component of a tree decomposition constructed for a vertex v.

First we consider bags of type σ. Let $e = \{v, w\} \in F$ such that v is its tail vertex and denote the corresponding σ-bag by X. Then, we make X the parent of the bag Y of type σ_T for the edge e. If v is the head vertex of e, then we make the bag Y of type σ_H for the edge e the parent of the bag X. As mentioned above, we do not create bags of type σ_H and σ_T for the two edges bounding the face with lowest layer number f_ℓ (for details see the proof of Lemma 6.12). Let $e_\ell \in \{e_{\ell_1}, e_{\ell_2}\}$. Then, we make the bag X of type σ corresponding to e_ℓ the parent of a bag Y of type σ_T corresponding to an edge e, if e and e_ℓ bound a face together, which is adjacent (in this case, sharing an edge) to

[(iv)] As opposed to the notation in the proof of Lemma 6.12, we use the vertex v as an argument for sets C as well to clarify that the faces we are considering in this step are incident faces of v.

the face f_ℓ. Analogously, we make Y the parent of X, if X is of type σ_H for such an edge e_ℓ.

Furthermore, we need to add edges between bags of types σ_T and σ_H as well. Note that by now, the only bag, which already has a parent is the bag of type σ_T for the unique incoming edge $e^* \in F$ in the spanning tree of G. We use the ordering $\mathrm{nb}_<(e, f)$ of the incident edges of a vertex v to make sure that the resulting tree decomposition is rooted. Let $\mathrm{nb}_\prec(e, f)$ express that two incident edges e, f of v are direct neighbors in the ordering $\mathrm{nb}_<(e, f)$. Suppose that X^* is the σ_T-bag for the edge e^* and Y is either a σ_H- or σ_T-bag for an edge f with either $\mathrm{nb}_\prec(e^*, f)$ or $\mathrm{nb}_\prec(f, e^*)$. In all of these cases, we make X^* the parent of Y, since X^* already has a parent bag. We observe that we have to direct the remaining edges in such a way that they point away from the bag X^*. Let $e, f \in \mathrm{Inc}(v) \setminus \{e^*, e_{\ell_1}, e_{\ell_2}\}$ with $\mathrm{nb}_\prec(e, f)$, X the σ_H/σ_T-bag of e and Y the σ_H/σ_T-bag of f. We have to analyze two cases. Note that always precisely one of the two holds.

(i) If $\mathrm{nb}_<(e^*, e)$, then make X the parent of Y.

(ii) If $\mathrm{nb}_<(f, e^*)$, then make Y the parent of X.

This completes existentially defining the tree decomposition as constructed in the proof of Lemma 6.12 in monadic second order logic for a 3-connected k-outerplanar graph.

We now count the parameters used in this proof. To find a face with lowest layer number for each vertex, we need the partition into its stripping layers as shown in Lemma 6.7. For this step we need k parameters. As explained above, for directing the edges of G we use $3k$ color sets (G has treewidth at most $3k - 1$ [4]) and one edge set (see Lemma 4.1). We fix edge sets for the spanning tree and the anchors $E_\mathcal{A}$ and co-anchors $E'_\mathcal{A}$ of the edge ordering $\mathrm{nb}_<(e, f)$. Hence, total number of parameters is $4k + 4$.

The predicates given in Appendix A.9.2 complete the proof. □

6.4 Implications of Hierarchical Graph Decompositions to Courcelle's Conjecture

By a well-known result, a connected graph G can be decomposed into its cut vertices and *blocks*, which then can be joined together in a tree structure (cf. Section 2.1 in [17]). These blocks are either single edges or maximal 2-connected subgraphs of G.[(v)] Hence, one can find a tree decomposition of a connected graph, such that each bag contains either a cut vertex one of its blocks, by making a bag containing a cut vertex v_c adjacent

[(v)] Let $G = (V, E)$ be a graph and $W \subseteq V$. $H = G[W]$ is called a *maximal 2-connected subgraph* of G, if for all $W' \supset W$, $G[W']$ is not 2-connected.

to each bag X containing a block of G, if $v_c \in X$. Clearly, in the general case, these tree decompositions do not have bounded width.

Analogously, Tutte showed that given a 2-connected graph (or a block of a connected graph) one can find a decomposition into its *2-cuts* and *3-blocks*, the latter of which are either 3-connected graphs or cycles (but not necessarily subgraphs of G, see below), which can be joined in a tree structure in the same way [30, Chapter 11] [31, Section IV.3]. Courcelle showed that both of these decompositions of a graph are MSOL-definable [14] and also proved that one can find an MSOL-definable tree decomposition of width 2, if all 3-blocks of a graph are cycles [14, Corollary 4.11]. In this section, we will use these methods to prove Courcelle's Conjecture for k-outerplanar graphs by showing that the results of the previous section can be applied to define tree decompositions of 3-connected 3-blocks of a k-outerplanar graph as well.

As many of our proofs make explicit use of the structure of Tutte's decomposition of a 2-connected graph into its 3-connected components, we will now review this concept more closely.

Definition 6.21 (3-Block)**.** Let $G = (V, E)$ be a 2-connected graph, $\mathcal{S}$ a set of 2-cuts of G and $W \subseteq V$. A graph $H = (W, F)$ is called a *3-block*, if it can be obtained by taking the induced subgraph of W in G and for each incident 2-cut $S = \{x, y\} \in \mathcal{S}$, adding the edge $\{x, y\}$ to F (if not already present), plus one of the following holds.

(i) H is a cycle of at least three vertices (referred to as a *cycle 3-block*).

(ii) H is a 3-connected graph (referred to as a *3-connected 3-block*).

Definition 6.22 (Tutte Decomposition)**.** Let $G = (V, E)$ be a 2-connected graph. A tree decomposition $(T = (N, F), X)$ is called a *Tutte decomposition* of G, if the following hold. Let $\mathcal{S}$ denote a set of 2-cuts of G.

(i) For each $t \in N$, X_t is either a 2-cut $S \in \mathcal{S}$ (called the *cut bags*) or the vertex set of a 3-block (called the *block bags*).

(ii) Each edge $f \in F$ is incident to precisely one cut bag.

(iii) Each cut bag is adjacent to precisely two block bags.

(iv) Let $t \in T$ denote a cut node with vertex set X_t. Then, t is adjacent to each block node t' with $X_t \subset X_{t'}$.

Tutte has shown that additional restrictions can be formulated on the choice of the set of 2-cuts, such that the resulting decomposition is unique for each graph (for details see the above mentioned literature). In the following, when we refer to *the* Tutte decomposition of a graph, we always mean the one that is unique in this sense, which is also the one that

Courcelle defined in his work [14]. Similarly, by a 3-connected 3-block (cycle 3-block, 2-cut etc.) of a graph G we mean a 3-connected 3-block in the Tutte decomposition of a block of G.

We will now state a property of Tutte decompositions, which will be useful in later proofs.

Definition 6.23 (Adhesion)**.** Let $(T = (N, F), X)$ be a tree decomposition. The *adhesion* of (T, X) is the maximum over all pairs of adjacent nodes $t, t' \in N$ of $|X_t \cap X_{t'}|$.

Proposition 6.24. *Each Tutte decomposition has adhesion 2.*

Proof. The claim follows directly from Definition 6.22 (ii) and (iv). □

For the proof of the next lemma, we need the notion of *W-paths*.

Definition 6.25. Let $G = (V, E)$ be a graph, $W \subseteq V$ and $x, y \in V$. Then, a path $P_{xy} = (V_P, E_P)$ between x and y is called a *W-path*, if $x, y \in W$ and $V_P \cap W = \{x, y\}$, i.e. P_{xy} avoids all vertices in W except its endpoints.

Lemma 6.26. *Let $G = (V, E)$ be a 2-connected graph with Tutte decomposition $(T = (N, F), X)$. If G is k-outerplanar, then all 3-connected 3-blocks $C = (W, F)$ of (T, X) are at most k-outerplanar.*

Proof. We know that $W = X_t$ for some $t \in N$. Let $S = \{x, y\}$ denote a 2-cut of G, which is incident to W. If $\{x, y\} \in E$, we do not have to consider S any further, so in the following, if we refer to a 2-cut S, we always assume that $\{x, y\} \notin E$. Since each such pair $\{x, y\}$ appears in precisely two 3-blocks (Definition 6.22 (iii)), we know that there is always at least one W-path between x and y in G.

Proposition 6.27. *Let $(T = (N, F), X)$ be a tree decomposition of adhesion 2 and $t \in T$. Let P_1 and P_2 denote two X_t-paths. If P_1 and P_2 share an internal vertex, then P_1 and P_2 have the same endpoints.*

Proof. Let $t \in N$. Then, all internal vertices of an X_t-path P are contained in a set of bags of a unique component T_t of $T[N \setminus \{t\}]$. Let $t' \in T_t$ be a neighbor of t. Then, the endpoints of P_1 and P_2 are contained in $X_t \cap X_{t'}$. Since (T, X) has adhesion 2, both paths have to have the same endpoints. □

Let $G' = G[W]$ denote the induced subgraph of G over the vertex set W. For each 2-cut S incident to W we add one W-path from G to G', connecting the two vertices in S. Since G is planar and G' is a subgraph of G, we know that G' is planar. Since (T, X) has adhesion 2 (Proposition 6.24), we know by Proposition 6.27 that there is no pair of W-paths corresponding to two different incident 2-cuts, sharing an internal vertex.

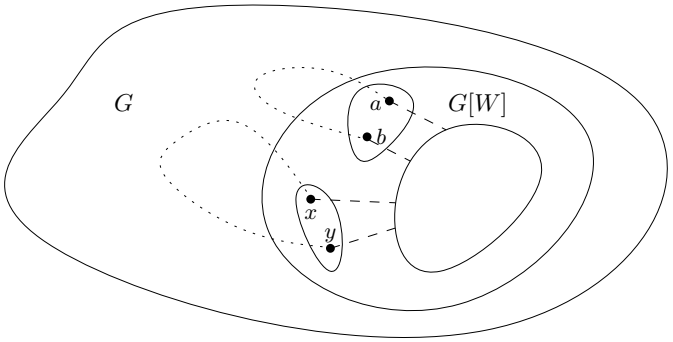


Figure 6.6: A 2-connected graph G with induced subgraph $G[W]$ over the vertex set of a 3-connected 3-block of G with incident 2-cuts $\{a, b\}$ and $\{x, y\}$. The dashed lines indicate that there might be several edges between a vertex and the depicted set and dotted lines represent (W-)paths in G.

Hence, we can contract each of these paths to a single edge such that the embedding of G' stays planar. Clearly, G' is isomorphic to C after contraction and the outerplanarity index of G' is less than or equal to k. □

For an illustration of the proof of Lemma 6.26, see Figure 6.6. The ideas in this proof can be applied to more general graph classes as well and we have the following consequence. For the proof of statement (ii), we need the following definition.

Definition 6.28 (Safe Separator [6])**.** Let $G = (V, E)$ be a connected graph with separator $S \subset V$. S is called a *safe separator*, if the treewidth of G is at most the maximum of the treewidth of all connected components W of $G[V \setminus S]$, by making S a clique in $G[W]$.

Corollary 6.29. *Let G be a 2-connected graph with Tutte decomposition (T, X).*

(i) If G is planar, then the 3-connected 3-blocks of (T, X) are planar.

(ii) If G is a partial k-tree, then the 3-connected 3-blocks of (T, X) are partial k-trees (for $k \geq 2$).

(iii) If G is $\mathcal{H}$-minor free, then the 3-connected 3-blocks of (T, X) are $\mathcal{H}$-minor free, where $\mathcal{H}$ is a set of fixed graphs.

Proof. (i) and (iii) follow from the same argumentation (and, clearly, (i) is a consequence of (iii) by Wagner's Theorem [33]). For (ii), we observe the following. By [14, Corollary 4.12] we know that each cut bag $S = \{x, y\}$ is a safe separator of G and hence, there is a width-k tree decomposition of G which has a bag X_{xy} containing both x and y. Subsequently, adding the edge between x and y does not increase the treewidth of a 3-connected 3-block B_3. (One simply performs a short case analysis of whether X_{xy} is contained in the tree decomposition of B_3 or not.) □

Replacing Edge Quantification by Vertex Quantification

As discussed above, a 3-block is in general not a subgraph of our graph G, as we add edges between the 2-cuts of the Tutte decomposition to turn the 3-blocks into cycles or 3-connected graphs. Since these absent edges can't be used as variables in MSOL-predicates (which would make our logic non-monadic), we need to find another way to quantify over them.

In [11], Courcelle discusses several structures over which one can define monadic second order logic of graphs, which we will now review.

Definition 6.30 (cf. 1.7 in [11])**.** Let $G = (V, E)$ be a graph. We associate with G two relational structures, denoted by $|G|_1 = \langle V, \text{edg} \rangle$ and $|G|_2 = \langle V \cup E, \text{edg}' \rangle$.

(i) All MSOL-sentences and -predicates over $|G|_1$ only use vertices or vertex sets as variables and we have that $\text{edg}(x, y)$ is true for $x, y \in V$, if and only if there is some edge $\{x, y\} \in E$. MSOL-sentences and -predicates over $|G|_2$ use both vertices and edges and vertex and edge sets as variables. Furthermore, $\text{edg}'(e, x, y)$ is true if and only if $e = \{x, y\}$ and $e \in E$.

(ii) If we can express a graph property in the structure $|G|_1$, we call it *1-definable* and if we can express a graph property in the structure $|G|_2$, we call it *2-definable*.

Clearly, the monadic second order logic we are using throughout this thesis is the one represented by the structure $|G|_2$. We use both vertex and edge quantification and one simply rewrites $\text{Inc}(v, e)$ to $\exists w \ \text{edg}'(e, v, w)$. Since every 1-definable property is trivially also 2-definable, we can conclude that both 1-definability and 2-definability imply MSOL-definability in our sense. Some of the main results of [11] can be summarized as follows.

Theorem 6.31 ([11])**.** *1-Definability equals 2-definability for*

(i) planar graphs.

(ii) partial k-trees.

(iii) $\mathcal{H}$-minor free graphs, where $\mathcal{H}$ is a set of fixed graphs.

Hence, by Theorem 6.31 we know that we can rewrite each formula using vertex and edge quantification to one only using vertex quantification, if a graph is a member of one of these classes. We will now show that this result can be used to implicitly quantify over virtual edges of a graph, if these virtual edges can be expressed by an (existentially) MSOL-definable relation. (For a similar application of this result, see [14, Problem 4.10].)

Lemma 6.32. *Let $G = (V, E)$ be a graph which is a member of a graph class $\mathcal{C}$ as stated in Theorem 6.31 and let P denote a graph property, which is 2-definable by a predicate ϕ_P. Let $E' \subseteq V \times V$ denote a set of virtual edges, such that there is a predicate $edg_{Virt}(v, w)$, which is true if and only if $\{v, w\} \in E'$. Then, P is 1-definable for the graph $G' = (V, E \cup E')$, if G' is a member of $\mathcal{C}$.*

Proof. By Theorem 6.31, P is 1-definable for the graph G. Let $\phi_{P|1}$ denote the predicate expressing P in $|G|_1$. We replace each occurrence of 'edg(x, y)' in $\phi_{P|1}$ by 'edg$(x, y) \vee$ edg$_{Virt}(x, y)$' and denote the resulting predicate by $\phi'_{P|1}$, which expresses the property P for the graph G' in $|G'|_1$. Since $G' \in \mathcal{C}$, one can replace quantification over sets of virtual edges (or mixed sets of edges and virtual edges) by vertex set quantification in the same way as for G. □

For the specific case of k-outerplanar graphs, we can now derive the following.

Corollary 6.33. *Let $G = (V, E)$ be a k-outerplanar graph and P a graph property, which is (C)MSOL-definable for 3-connected k-outerplanar graphs. Let B_3 denote a 3-block of G, including the virtual edges between all incident 2-cuts of B_3. Then, P is (C)MSOL-definable for B_3.*

Proof. By [14, Section 3] we know that there is a predicate $\phi_{\mathcal{C}_2}(x, y)$, which is true, if and only if $\{x, y\}$ is a 2-cut in the Tutte decomposition of (a block of) G. We know that B_3 (including the virtual edges) is still k-outerplanar (Lemma 6.26). Hence let edg$_{Virt}(x, y) = \phi_{\mathcal{C}_2}(x, y)$ and apply Lemma 6.32. □

Note that the statements of Lemma 6.32 and Corollary 6.33 also hold for existential definability.

Defining the Tree Decomposition of a k-Outerplanar Graph

By Corollary 6.33 we now know that every graph property, which can be defined for a 3-connected k-outerplanar graph, can also be defined for a 3-block of any k-outerplanar graph G (including its virtual edges).

To apply these results to any k-outerplanar graph G, we first show how to construct an existentially definable tree decomposition of G, assuming that there exist predicates existentially defining bounded width tree decompositions for the 3-connected 3-blocks of (the Tutte decomposition of the 2-blocks of) G. For an illustration of the proof idea of the following Lemma, see Figure 6.7, which shows that we can fix a parent-child ordering of the hierarchical graph decomposition of G. After replacing the 3-blocks of G by their corresponding tree decompositions (taking into account the direction of the

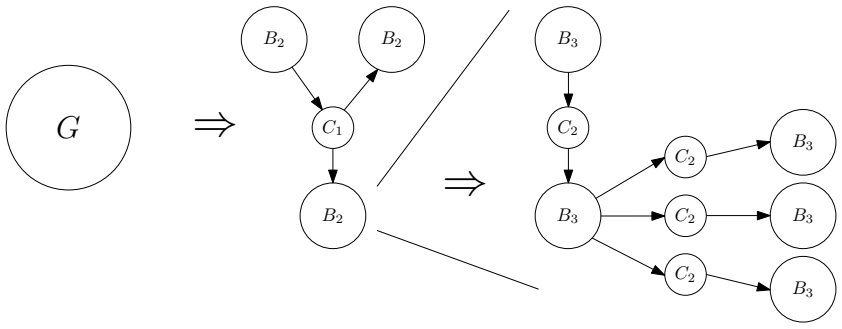


Figure 6.7: An example hierarchical decomposition of a graph G. A bag labeled C_1 contains a cut-vertex of G, C_2 a 2-cut of G. Bags labeled B_2 contain a 2-block (a single edge or a maximal 2-connected component). If a 2-block contains a maximal 2-connected component of G, it is decomposed further into its 2-cuts and 3-blocks, labeled by B_3, which contain either a cycle or a 3-connected 3-block.

edges in the hierarchical decomposition), one can see that we have a bounded width tree decomposition of the entire graph G.

Remark 6.34. Note that in the proofs of the following results, one guesses a rooted spanning tree $S = (V, F_S)$ of a k-outerplanar graph $G = (V, E)$, which will be used to induce a parent-relation on the bags of the hierarchical decomposition of G (see Figure 6.7). The edges of this spanning tree will also be used to induce a parent-relation in the tree decompositions of the 3-connected 3-blocks of G (see Lemma 6.39) and hence we know that we can replace each bag containing a 3-connected 3-block in a Tutte decomposition by its tree decomposition without creating a conflicting parent-child ordering in the thus resulting tree decomposition of G.

Lemma 6.35. *Let $G = (V, E)$ be a k-outerplanar graph with Tutte decompositions (T, X) of its 2-connected blocks. Then, G admits an existentially MSOL-definable tree decomposition of width at most $3(k+1)$ with a constant number of parameters, if there exist predicates existentially defining width-$3k$ tree decompositions for the 3-connected 3-blocks of G with a constant number of parameters.*

Proof. Recall the decomposition of a graph into its 3-connected components described in the beginning of Section 6.4 and see Figure 6.7 for an illustration. We will first show how to construct a rooted tree decomposition $(\mathcal{T} = (\mathcal{N}, \mathcal{F}), \mathcal{X})$ of G width at most $3k+3$ and then prove that $(\mathcal{T}, \mathcal{X})$ is indeed MSOL-definable. Naturally, the description of the tree decomposition is already aimed at providing straightforward methods to define its predicates in MSOL.

I. Constructing the tree decomposition. We use the following notation. $\mathcal{C}_1$ denotes the set of singletons containing a cut-vertex of G and $\mathcal{C}_2$ denotes the set of 2-cuts in all Tutte decompositions of the 2-connected blocks of G. Furthermore, $\mathcal{B}_2$ denotes the set of blocks of G, $\mathcal{B}_2^E$ the set of blocks that are single edges and $\mathcal{B}_3$ denotes the set of 3-blocks of (T, X). Let $\Theta_{\mathcal{B}_3} = \{\Theta_1, \ldots, \Theta_r\}$ denote the set of tree decompositions of all

elements in $\mathcal{B}_3$. Then, we create a bag in $(\mathcal{T}, \mathcal{X})$ for all elements in $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{B}_2^E$ and all bags of each Θ_i in $\Theta_{\mathcal{B}_3}$, where $1 \leq i \leq r$. Note that if a 3-block $B_3 \in \mathcal{B}_3$ is a cycle, one can find a tree decomposition of B_3 of width 2 directly. We will later study how to find an MSOL-definable tree decomposition of such a cycle in a more detailed way.

(In the following, keep Remark 6.34 in mind.) We add an edge to $\mathcal{F}$ between all pairs of adjacent bags originating from a tree decomposition Θ_i with the same orientation. To make $\mathcal{T}$ a directed tree, we add edges to $\mathcal{F}$ between the above mentioned components in the following way. First, we take a spanning tree $S = (V, F_S)$ of G with an arbitrary root $r \in V$. For each vertex $x \in V$, let P_x denote the path from r to x in S.

Let $B_2 \in \mathcal{B}_2 \setminus \mathcal{B}_2^E$ with Tutte decomposition $(T = (N, F), X)$. We know that the bags of (T, X) either contain a 2-cut $C_2 \in \mathcal{C}_2$ or a 3-block $B_3 \in \mathcal{B}_3$ with tree decomposition $\Theta_i \in \Theta_{\mathcal{B}_3}$ for some i with $1 \leq i \leq r$. We now show which edges we need to add to $\mathcal{F}$ and how to direct them to obtain a tree decomposition of B_2 of width at most $3k + 2$. We know that each edge in F is incident to one cut bag and one block bag (Definition 6.22(ii), cf. Figure 6.7). Let C_2, B_3 and Θ_i be as above and additionally $C_2 \subset B_3$. By Definition 6.22(iv) we know that there has to be an edge in $\mathcal{F}$ between C_2 and one bag in Θ_i, as there is an edge in F between C_2 and B_3. Since C_2 is a separator of G we know that for all $v \in B_3$, $v \neq x$ and $v \neq y$ one of the following cases holds. (Either all paths P_v go through x or y or none of them.)

(i) $P_x \sqsubset P_v$ or $P_y \sqsubset P_v$.

(ii) $P_v \sqsubset P_x$ or $P_v \sqsubset P_y$.

In case (i), we add the vertices x and y to all bags in Θ_i and add an edge between $\mathcal{X}_t$ and the root of Θ_i. Hence, we introduce at most two vertices to each bag. Referring to Figure 6.7, this is the case when the cut bag is the parent of a block bag. By Definition 6.22(iii) there is precisely one such bag.

However, a block bag can be the parent of an unbounded number of cut bags, which is dealt with in case (ii). Since there is a (virtual or non-virtual) edge between the vertices x and y in B_3, we know that there exists at least one bag in Θ_i containing the two vertices. We need to find one directly identifiable bag $\mathcal{X}_{t'}$ in Θ_i, which we can make a parent of $\mathcal{X}_t$. If there is a bag X^* in Θ_i containing x and y, whose parent bag does not contain both vertices, then we let $\mathcal{X}_{t'} = X^*$. If no such X^* exists, then we let $\mathcal{X}_{t'}$ be the root of Θ_i, which subsequently has to contain both x and y. One can verify that this yields a rooted tree decomposition of width at most $3k + 2$ for any $B_2 \in \mathcal{B}_2 \setminus \mathcal{B}_2^E$.

To finish the construction of the rooted tree decomposition $(\mathcal{T}, \mathcal{X})$, we need to show, which edges to add to $\mathcal{F}$ between bags in $\mathcal{C}_1$ and (tree decompositions of elements in) $\mathcal{B}_2$. We use the same idea as before, based on the spanning tree S of G. In the following

let $C_1 = \{x\} \in \mathcal{C}_1$ and $B_2 \in \mathcal{B}_2$ with $C_1 \subset B_2$. We have the same two cases as before. Since C_1 is a separator of G, one of the following holds for all $v \in B_2$, $v \neq x$.

(i) $P_x \sqsubset P_v$.

(ii) $P_v \sqsubset P_x$.

Again, in case (i), there is a bag $\mathcal{X}_t$ with $\mathcal{X}_t = C_1$, which is the parent bag of some bag associated with the component B_2. We add x to all bags associated with B_2 and make $\mathcal{X}_t$ the parent of a bag $\mathcal{X}_{t'}$, where $\mathcal{X}_{t'}$ is a bag with $\mathcal{X}_{t'} = B_2$ in case $B_2 \in \mathcal{B}_2^E$ and if $B_2 \in \mathcal{B}_2 \setminus \mathcal{B}_2^E$, $X_{t'}$ is the root bag of the tree decomposition of B_2, constructed as described above. In case (ii) when $B_2 \in \mathcal{B}_2^E$, we simply let the bag $\mathcal{X}_t$ with $\mathcal{X}_t = B_2$ be the parent of the bag $\mathcal{X}_{t'}$ with $\mathcal{X}_{t'} = C_1$. If $B_2 \in \mathcal{B}_2 \setminus \mathcal{B}_2^E$, we observe the following. Since x is a cut vertex of G, no 2-cut of a block of G can contain x. Hence we know that there exists one unique 3-block $B_3^* \in \mathcal{B}_3$ with $x \in B_3^*$. We denote its tree decomposition by Θ_i^*. Again, we find a bag $\mathcal{X}_t$ in Θ_i^*, such that its parent does not contain x. If no such bag exists, we let $\mathcal{X}_t$ be the root of Θ_i^*. We again let $\mathcal{X}_{t'}$ be the bag with $\mathcal{X}_{t'} = C_1$ and make $\mathcal{X}_t$ the parent of $\mathcal{X}_{t'}$.

One can verify that now $(\mathcal{T}, \mathcal{X})$ is a rooted tree decomposition and since in the last stage we introduced at most one vertex to each bag of a tree decomposition of an element in $\mathcal{B}_2$, its width is at most $3k + 3$.

II. Definability. For defining all necessary predicates for the tree decomposition $(\mathcal{T}, \mathcal{X})$, we will refer to G as the graph after adding all virtual edges of its Tutte decomposition. We might write down predicates quantifying over virtual edges or having virtual edges as free variables, and by Corollary 6.33 we know that all these predicates can be defined only using vertex quantification as well.

By some trivial definitions, the statement of the lemma, and the results of [14] we know that the predicates listed below exist.

Proposition 6.36 (cf. [14])**.** *Let $G = (V, E)$ be a k-outerplanar graph, for whose 2-blocks all Tutte decompositions are known. Let $G' = (V, E \cup E')$ denote the graph obtained by adding all corresponding virtual edges E' to G and $\gamma : V \to \mathbb{N}_{|3k+1}$ a coloring of V in G'. The following predicates are MSOL-definable.*

(I) $Bag_{\mathcal{C}_1}(v, X)$*: $X \in \mathcal{C}_1$ and $X = \{v\}$.*

(II) $Bag_{\mathcal{B}_2^E}(e, X)$*: $X \in \mathcal{B}_2^E$ and $X = \{v, w\}$, where $e = \{v, w\}$.*

(III) $2\text{-}Conn_{\mathcal{B}_2 \setminus \mathcal{B}_2^E}(X)$*: X is the vertex set of a 2-connected 2-block of G.*

(IV) $Bag_{\mathcal{C}_2}(v, X)$*: $X \in \mathcal{C}_2$, $v \in X$ and for $w \in X$, $v \neq w$, we have $\gamma(v) < \gamma(w)$.*

(V) $3\text{-}Conn_{\mathcal{B}_3}(X)$*: X is the vertex set of a 3-connected 3-block of G.*

(VI) $Cycle_{\mathcal{B}_3}(X)$*: X is a set of vertices forming a cycle block in a 2-block of G.*

(VII) $Bag^{\mathcal{B}_3}_{\tau_1}(v,X),\ldots,Bag^{\mathcal{B}_3}_{\tau_t}(v,X), Bag^{\mathcal{B}_3}_{\sigma_1}(e,X),\ldots,Bag^{\mathcal{B}_3}_{\sigma_s}(e,X)$*: The Bag-predicates of the tree decompositions of the 3-connected 3-blocks of G.*

(VIII) $Parent_{\mathcal{B}_3}(X,Y)$*: The Parent-predicate of the tree decompositions of the 3-connected 3-blocks of G.*

Proof. (I) and (II) follow from [14, Lemma 2.1], (III) from [14, Section 2] and (IV) from [14, Section 3] and Corollary 6.33. (V) is shown in [14, Corollary 4.8] and a proof of (VI) can done with the same argument. Finally, (VII) and (VIII) are part of the statement of the lemma. □

We now turn to defining tree decompositions for the cycle 3-blocks of a graph, after which we only need to show that gluing together all components of our construction explained above is MSOL-definable.

Proposition 6.37. *Let $G=(V,E)$ be a graph and $C=(W,F)$ a cycle 3-block of G (including virtual edges). There is an existentially definable predicate $Bag_{Cyc}(e,X)$, which is true if and only if X is a bag of a tree decomposition of C associated with a (possibly virtual) edge e and an existentially definable predicate $Parent_{Cyc}(X,Y)$ encoding a parent-relation of a tree decomposition of C.*

Proof. Recall that for orienting the edges of our tree decomposition, we first find a spanning tree S of the graph G with root r and note that by Proposition 6.36(V), W is MSOL-definable. To create a definable tree decomposition of C, we now find a root $r_C \in W$ of C. If $r \in W$, we let $r_C = r$, otherwise we know that there is one parent component $C_P \in \mathcal{C}_1 \cup \mathcal{C}_2$ of C in G. C_P can be identified by checking for all 1- and 2-cuts C_C, which are incident to W, if all paths in S from r to the vertices $w \in W$ pass through (at least one of the vertices in) C_C. This can be defined in a straightforward way and one can see that there is always precisely one such cut. If $C_P = \{x\} \in \mathcal{C}_1$, then we let $r_C = x$ and if $C_P = \{x,y\} \in \mathcal{C}_2$, then we let $r_C = x$, if $\gamma(x) < \gamma(y)$ in a fixed coloring γ of C. We create a bag X for each edge $f=\{v,w\} \in F$, which is not incident to r_C and let $X = \{r_C, v, w\}$. Hence, the predicate $\mathrm{Bag}_{Cyc}(e,X)$ is also definable in a straightforward way.

We then orient the edges in F in such a way that C is a directed cycle. Note that one can find a conflict-free ordering for all cycle blocks in the graph G. (Otherwise, we might violate the cardinality constraint of MSOL.) The predicate $\mathrm{Parent}_{Cyc}(X,Y)$ is true, if and only if the following hold.

(i) There are two edges $e, f \in F$, such that $\mathrm{Bag}_{Cyc}(e,X)$ and $\mathrm{Bag}_{Cyc}(f,Y)$ (and e and f are contained in the same cycle).

(ii) The directed path from r_C to tail(e) in C is a strict subpath of the path from r_C to tail(f).

(iii) $|X \cap Y| = 2$.

Note that we only need one additional parameter, the edge set defining the edge orientation of F, since we already have a coloring for the entire graph G (see Proposition 6.36). The details of the predicates in Appendix A.10.1 complete the proof. □

To unify the parent-relations for all tree decompositions of 3-blocks, we can write

$$\text{Parent}'_{\mathcal{B}_3}(X, Y) \Leftrightarrow \text{Parent}_{\mathcal{B}_3}(X, Y) \vee \text{Parent}_{Cyc}(X, Y).$$

As described above, to create the according parent-relation between blocks of the hierarchical decomposition of G, we need to add a number of vertices to some of the bags of the final tree decomposition $(\mathcal{T}, \mathcal{X})$. The details for the changes in those definitions are presented in Appendix A.10.2. We can define a Parent-predicate for $(\mathcal{T}, \mathcal{X})$ by using the ideas explained above to add edges between blocks and cut-bags. Let $\text{Parent}_{\mathcal{BC}}(X, Y)$ denote such a predicate. Then, we have that

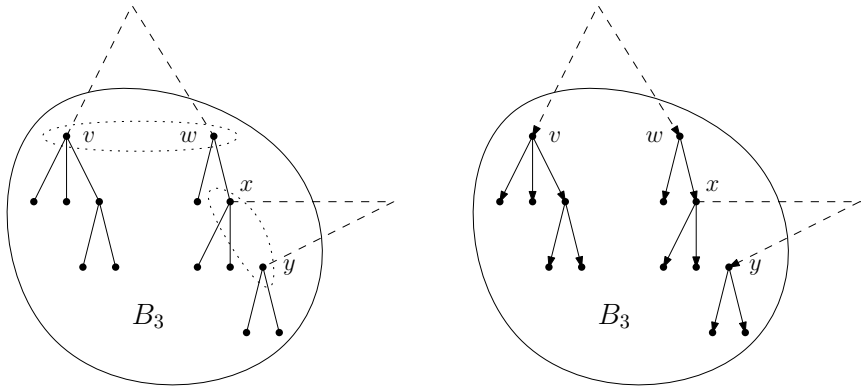
$$\text{Parent}(X, Y) \Leftrightarrow \text{Parent}'_{\mathcal{B}_3}(X, Y) \vee \text{Parent}_{\mathcal{BC}}(X, Y).$$

To show that the number of parameters that we need to define the above mentioned predicates is constant, we note that we only use constructions of previous results with constant numbers of parameters. (For the exact number see the corresponding result.) Note that for the cycle components one additional parameter is as well enough (cf. the proof of Proposition 6.37) to turn all cycles into directed cycles, since they are connected in a tree structure in the Tutte decomposition of G. Hence, fixing the direction of one cycle will always yield the possibility to direct adjacent (i.e. sharing a 2-cut) cycles in a conflict-free manner.

The details for the predicate $\text{Parent}_{\mathcal{BC}}(X, Y)$ are given in Appendix A.10.2 and complete the proof of Lemma 6.35. □

As mentioned in the previous proof, another obstacle in applying Lemma 6.19 to define a tree decomposition for G using its (definable) hierarchical graph decomposition is the cardinality constraint of MSOL. We illustrate this problem with an example.

Example 6.38. Let $G = (V, E)$ be a k-outerplanar graph with $\mathcal{O}(n/\log n)$ 3-connected 3-blocks of size $\mathcal{O}(\log n)$. Let P denote a graph property, which is definable for 3-connected k-outerplanar graphs by a predicate ϕ_P. Suppose that ϕ_P uses a constant number of parameters. When applying ϕ_P to all 3-connected 3-blocks of G, this might result in a predicate using $\mathcal{O}(n/\log n)$ parameters and hence, P not definable in this straightforward way for G.



(a) T_{B_3} without edge direction.

(b) T_{B_3} with edge direction.

Figure 6.8: A forest T_{B_3} of a 3-connected 3-block of an example graph. The dashed lines indicate the paths in T between two endpoints of a incident cut of B_3. Here, $\{v, w\}$ is the root cut of B_3 and $\{x, y\}$ a child cut. Note that by Propositions 6.42 and 6.43, this small example is already somewhat general.

However, for the case of defining a tree decomposition of a k-outerplanar graph, we can avoid this problem. When defining a tree decomposition for a 3-connected k-outerplanar graph in MSOL, one first guesses a rooted spanning tree of G. To avoid guessing a non-constant number of spanning trees, we will find a set of edges $\mathcal{S}_E$, which contains a spanning tree with bounded edge and face remember number for each 3-connected 3-block of G. Furthermore we guess one set $\mathcal{R}_V$, containing one unique vertex for each 3-connected 3-block of G, which we will use as the root of its spanning tree. We need to make some observations about such candidate sets $\mathcal{S}_E$ and $\mathcal{R}_V$. We first prove the existence of these sets and then their MSOL-definability.

Lemma 6.39. *Let $G = (V, E)$ be a planar graph and $G = (V, E \cup E')$ the graph obtained by adding the virtual edges E' of the Tutte decompositions of the 2-connected blocks of G to G. Let $T = (V, F)$ be a spanning tree of G with $er(G, T) \leq \lambda$ and $fr(G, T) \leq \mu$. Let $B_3 = (V_{B_3}, E_{B_3}) \in \mathcal{B}_3$ be a 3-connected 3-block of G' (including virtual edges) and $T_{B_3} = T[V_{B_3}]$. One can construct from T_{B_3} a spanning tree $T^*_{B_3}$ of B_3 with $er(B_3, T^*_{B_3}) \leq \lambda$ and $fr(B_3, T^*_{B_3}) \leq \mu$ by adding edges from $E \cup E'$ to T_{B_3}.*

Proof. Clearly, $T_{B_3} = (V_{B_3}, F_{B_3})$ is a forest in B_3 and in the following we denote its tree components by $F_1 = (V_{F_1}, E_{F_1}), \ldots, F_c = (V_{F_c}, E_{F_c})$. We will now show how to connect these components to a tree. Let $\mathcal{C}'_2 \subseteq \mathcal{C}_2$ denote the set of incident 2-cuts of B_3.

Proposition 6.40. *Let T'_{B_3} denote the graph obtained by adding an edge between all 2-cuts $\{x, y\} \in \mathcal{C}'_2$ in T_{B_3} (if not already present). Then, T'_{B_3} is connected.*

Proof. Let $(T_T = (N_T, F_T), X)$ denote the Tutte decomposition containing B_3 and let $B_3 = X_t$ with $t \in N_T$. Let $v, w \in V_{B_3}$ and consider the unique path P_{vw} between v and

w in T. There are two cases: (I) The path P_{vw} is completely contained in B_3 and v and w belong to the same connected component. (II) Suppose that they don't and let F_i denote the component with $v \in V_{F_i}$ and F_j the component with $w \in V_{F_j}$. Let x and y be the vertices on the path P_{vw} with $x, y \in V_{B_3}$ (and $x \neq y$), such that x has a neighbor $x' \notin V_{B_3}$ and y has a neighbor $y' \notin V_{B_3}$ (both in P_{vw}). Denote this subpath by P_{xy}. Then, P_{xy} is a V_{B_3}-path in G. Hence, there is a unique component in $T'_T = T_T[N_T \setminus \{t\}]$ containing all internal vertices of P_{xy}. Since the neighbor of t in T'_T is a cut-bag, we know that it has to contain both x and y and hence $\{x, y\} \in \mathcal{C}'_2$. □

By Proposition 6.40 we know that we can find a subset of incident 2-cuts of each 3-connected 3-block to turn T_{B_3} into a tree. We now prove that adding these edges does not increase the edge and face remember number. Consider a 2-cut $C_2 = \{x, y\} \in \mathcal{C}'_2$, such that $\{x, y\} \notin F$. Since T is a spanning tree of G, we know that there is one unique path P_{xy} between x and y in T. Let $T'_{B_3} = (V'_{B_3}, F'_{B_3})$ denote the tree obtained by adding the above described paths between the components of T_{B_3}. Then, T'_{B_3} is a spanning tree of the graph $G'_{B_3} = (V'_{B_3}, E_{B_3} \cup F'_{B_3})$ with $er(G'_{B_3}, T'_{B_3}) \leq \lambda$ and $fr(G'_{B_3}, T'_{B_3}) \leq \mu$, since $G'_{B_3} \sqsubseteq G$ and no edges, which are not members of T'_{B_3}, are introduced in G'_{B_3}. Subsequently, replacing each path P_{xy} by a single edge in T'_{B_3} does not increase the edge and face remember number as well and after these replacements, we have that $T'_{B_3} = T^*_{B_3}$ and our claim follows. For an illustration of this proof see Figure 6.8a. □

Lemma 6.41. *The statement of Lemma 6.39 also holds, if one replaces the term* spanning tree *by* rooted spanning tree*. Furthermore there is a set $\mathcal{R}_V \subseteq V$, which contains precisely one vertex acting as a root for a spanning tree for each 3-connected 3-block of G.*

Proof. We use the same notation as in the proof of Lemma 6.39. Since $T = (V, F)$ is a rooted spanning tree, we know that its components $F_1, \ldots, F_c$ in B_3 are rooted trees as well, see Figure 6.8b for an illustration. Since the direction between block and cut bags of a Tutte decomposition of a block of G are based on the spanning tree T (see Remark 6.34 and the proof of Lemma 6.35), we observe the following. Let $C_2 = \{x, y\} \in \mathcal{C}_2$ denote an incident 2-cut of B_3 with $\{x, y\} \notin F$. There are two cases we have to consider. Either, C_2 is the parent cut of B_3 or it is a child cut.

Proposition 6.42. *Let C_2 be a child cut of B_3. Wlog. x is a vertex in a tree F_i and y is the root of a tree F_j.*

Proof. Suppose not. We know that there is a path P_{xy} between x and y in T. If y is a non-root vertex in F_j, then we can't direct the edges of P_{xy} in T such that every vertex has precisely one parent. Hence, T is not a directed tree and we have a contradiction. □

Proposition 6.43. *Let C_2 be the parent cut of B_3. Then, x and y are roots of two trees F_i and F_j.*

Proof. For any vertex $v \in V_{B_3}$ we know by definition (see the proof of Lemma 6.35) that for every vertex $v \in V_{B_3}$, the directed path from the root r of T to v in T is either a subpath of the directed path from r to x or from r to y. Hence, neither x nor y can have a parent in T_{B_3}. □

We can direct the additional edges by using Propositions 6.42 and 6.43. In the case that C_2 is a child cut, we can always direct the edge $\{x, y\}$ from x to y (using the notation of Proposition 6.42). If C_2 is the parent cut, we know by Proposition 6.43 that we can orient $\{x, y\}$ arbitrarily. There are two cases we need to analyze to make sure we do not create a conflicting orientation of $\mathcal{S}_E$. In the first case, the edge $\{x, y\}$ has been added to $\mathcal{S}_E$ by the parent block of C_2. We then use the same orientation. In the second case, if $\{x, y\} \notin \mathcal{S}_E$, we can choose the direction arbitrarily.

We now turn to finding the set of roots $\mathcal{R}_V$. If B_3 is the root block according to the spanning tree of G with root r_G, then we add r_G to $\mathcal{R}_V$ as the root of B_3. Otherwise, we find its parent cut $C_2 = \{x, y\}$. Assume wlog. that the edge $\{x, y\}$ is directed from x to y according to the construction explained above. Then we add x to $\mathcal{R}_V$. Since each cut-bag has precisely one child block bag (Definition 6.22(ii)), we know that this vertex is unique for each 3-block B_3. □

Lemma 6.44. *The sets $\mathcal{S}_E$ and $\mathcal{R}_V$ of Lemmas 6.39 and 6.41 are existentially MSOL-definable with $3k + 2$ parameters.*

Proof. Let $G = (V, E)$ denote a k-outerplanar graph, such that the virtual edges introduced by the Tutte decompositions of its 2-connected blocks are already included in E. On a high level, for defining $\mathcal{R}_V$ and $\mathcal{S}_E$, we need to encode is the following:

(i) There are sets $\mathcal{R}_V \subseteq V$, $F \subseteq E$ and $F' \subseteq E$ with $\mathcal{S}_E = F \cup F'$.

(ii) Guess a root $r_T \in V$, such that F is the edge set of a rooted spanning tree in G.

(iii) An edge $e = \{x, y\}$ is possibly (but not necessarily) a member of F', if $\{x, y\} \in \mathcal{C}_2$ and $e \notin F$.

(iv) For all $B_3 \in \mathcal{B}_3$, the graph $T^*_{B_3} = (B_3, \mathcal{S}_E \cap (B_3 \times B_3))$ is a spanning tree of the graph $G_{B_3} = G[B_3]$ with $er(G_{B_3}, T^*_{B_3}) \leq 2k$ and $fr(G_{B_3}, T^*_{B_3}) \leq k$.

(v) A vertex $v \in V$ is possibly (but not necessarily) a member of $\mathcal{R}_V$, if it is a member of a 2-cut $\{v, w\} \in \mathcal{C}_2$.

(vi) For each 3-connected 3-block $B_3 \in \mathcal{B}_3$, there is a vertex $r_{B_3} \in \mathcal{R}_V$, such that $T^*_{B_3}$ can be rooted at r_{B_3} (without altering the edge direction of any other edge in $\mathcal{S}_E$).

The existence of such sets $\mathcal{R}_V$ and $\mathcal{S}_E$ is shown in Lemmas 6.39 and 6.41, so we do not need to encode all details mentioned in the corresponding proofs explicitly. Property (iv) is MSOL-definable by Proposition 6.20, since G_{B_3} is 3-connected. We have the following parameters. First, the edge set of the spanning tree and again a $3k$-coloring and one edge set to fix the orientation of the edges in $\mathcal{S}_E$.

The details of the predicates encoding the rest of the properties are given in Appendix A.10.3 and complete the proof. □

We can now use the above results to conclude that we can find predicates defining tree decompositions of 3-connected 3-blocks of k-outerplanar graphs.

Corollary 6.45. *Let $G = (V, E)$ be a k-outerplanar graph. Then, there exist predicates existentially defining tree decompositions of width at most $3k$ for each 3-connected 3-block of G with a constant number of parameters.*

Proof. By Lemma 6.19 we know that a 3-connected k-outerplanar graph admits an MSOL-definable tree decomposition of width $3k$, based on a rooted spanning tree of the graph. By Corollary 6.33 we can define such a tree decomposition in a structure, which also includes the virtual edges of a 3-block in G (and by Lemma 6.26 we know that this graph is still k-outerplanar). Finally, by Lemmas 6.39, 6.41 and 6.44 we know that we can find definable edge and vertex sets which contain the edges of spanning trees for each 3-connected 3-block with the required bound on their vertex and edge remember numbers without violating the cardinality constraint of monadic second order logic. Similarly, we can find sets containing anchor and co-anchor edges for all 3-connected 3-blocks in a straightforward way. Hence, also for defining the ordering of all incident edges of all vertices in a 3-connected 3-block, two sets are sufficient. Subsequently, the number of parameters involved is bounded by a constant. For the exact bounds see the corresponding result. □

Combining Lemma 6.35 and Corollary 6.45 with Theorem 5.12 yields the main result of this chapter.

Theorem 6.46. *CMSOL-definability equals recognizability for k-outerplanar graphs.*

7

Conclusion

In this thesis, we investigated a conjecture by Courcelle from 1990, which states that every recognizable graph property is also definable in counting monadic second order logic [9]. We introduced a new proof technique, based on the Myhill-Nerode theory for graphs of bounded treewidth [18, Section 12.7] (Theorem 2.18), to give self-contained proofs of a number of special cases for this conjecture. Theorem 2.18 states that recognizability equals finite index for graphs of bounded treewidth and we have shown that for any graph class, which admits MSOL-definable tree decompositions, finite index implies definability in (counting) monadic second order logic. We do not need the counting operation of CMSOL, if the tree decomposition of a graph class is either ordered or has bounded degree. We showed how to construct bounded degree MSOL-definable tree decompositions for Halin graphs and some subclasses of k-outerplanar graphs (including bounded degree and 3-connected k-outerplanar graphs). For all of these graph classes (in particular for bounded degree graphs or 3-connected planar graphs), one can find an MSOL-definable linear ordering and Courcelle showed that in this case, CMSOL-definability equals MSOL-definability [13] (see also [3]). Halin graphs are known to have treewidth 3 [35] and Kaller proved that recognizability implies CMSOL-definability for graphs of treewidth at most 3 [22], so combining these results with the above mentioned orderability of 3-connected planar graphs already yields our result that recognizability implies MSOL-definability for the case of Halin graphs. These observations raise the following interesting question.

Question 1. Is it possible to construct an ordered or bounded degree MSOL-definable tree decomposition for a graph class that does not admit an MSOL-definable linear ordering?

These two notions seem very closely intertwined. Bags in a tree decomposition are associated with vertices or edges in the graph and one can use a linear ordering to either order all children of a bag or create a tree decomposition, whose bags have bounded

degree (cf. Lemma 6.19). Even though the other direction of this argument might not hold in general,[(i)] such a graph class could still be hard to find.

The main result of this thesis is that recognizability implies CMSOL-definability for the class of k-outerplanar graphs. We first showed how to construct a bounded degree MSOL-definable tree decomposition for 3-connected k-outerplanar graphs and then used results about hierarchical graph decompositions (decomposing a connected graph into its 2- and then 3-connected components) to define tree decompositions for all k-outerplanar graphs. We gave some hints (see e.g. Corollary 6.29 and Lemma 6.32) that this technique might be used in other proofs of (special cases of) Courcelle's Conjecture as well. As 3-connected graphs have favorable properties when it comes to results about MSOL-definability (see e.g. the above mentioned linear orderings for 3-connected planar graphs or Proposition 6.14), we believe a proof that Courcelle's Conjecture holds for graph classes, whose 3-connected members admit MSOL-definable tree decompositions might be an important step towards its full resolution. Naturally, we impose the following question.

Question 2. Let $\mathcal{C}$ be a graph class and $\mathcal{C}_{|3C}$ denote all its 3-connected members. Does resolving Courcelle's Conjecture for all members of $\mathcal{C}_{|3C}$ imply its resolution for all members in $\mathcal{C}$?

As one can see in the proofs of Section 6.4, this question is not easy to answer, since we can only apply results of 3-connected members of a graph class to 3-connected components of a graph, if we can define predicates, which only require a constant amount of parameters for an entire graph. Depending on how one constructs a tree decomposition for a graph class, this might be a major obstacle in resolving Question 2 for a specific graph class $\mathcal{C}^*$.

[(i)] I.e., one can always derive a linear ordering of a graph G given a bounded degree or ordered MSOL-definable tree decomposition of G.

Bibliography

[1] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth, and well-quasi-ordering for bounded treewidth. In *Proceedings of the AMS Summer Workshop on Graph Minors and Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 539–564. AMS, 1993.

[2] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[3] A. Blumensath and B. Courcelle. Monadic second-order definable graph orderings. *Logical Methods in Computer Science*, 10(2), 2014.

[4] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[5] H. L. Bodlaender, P. Heggernes, and J. A. Telle. Recognizability equals definability for graphs of bounded treewidth and bounded chordality. In *Proceedings EUROCOMB 2015*, Electronic Notes in Discrete Mathematics. Elsevier, 2015.

[6] H. L. Bodlaender and A. M. Koster. Safe separators for treewidth. *Discrete Mathematics*, 306(3):337 – 350, 2006.

[7] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(1-6):555–581, 1992.

[8] J. R. Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[9] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[10] B. Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991.

[11] B. Courcelle. The monadic second order logic of graphs VI: On several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54(23):117 – 149, 1994.

[12] B. Courcelle. The monadic second-order logic of graphs VIII: Orientations. *Annals of Pure and Applied Logic*, 72(2):103–143, 1995.

[13] B. Courcelle. The monadic second-order logic of graphs X: Linear orderings. *Theoretical Computer Science*, 160(12):87 – 143, 1996.

[14] B. Courcelle. The monadic second-order logic of graphs XI: Hierarchical decompositions of connected graphs. *Theoretical Computer Science*, 224(12):35 – 58, 1999.

[15] B. Courcelle. The monadic second-order logic of graphs XII: Planar graphs and planar maps. *Theoretical Computer Science*, 237(12):1 – 32, 2000.

[16] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach*, volume 138 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2012.

[17] R. Diestel. *Graph Theory*. Number 173 in Graduate Texts in Mathematics. Springer, 4 edition, 2012. Corrected reprint.

[18] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[19] R. Halin. Studies on minimally n-connected graphs. *Combinatorial Mathematics and its applications*, pages 129–136, 1971.

[20] R. Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[21] V. Kabanets. Recognizability equals definability for partial k-paths. In *Proceedings ICALP 1997*, volume 1256 of *LNCS*, pages 805–815. Springer, 1997.

[22] D. Kaller. Definability equals recognizability of partial 3-trees and k-connected partial k-trees. *Algorithmica*, 27(3-4):348–381, 2000.

[23] I. Katsikarelis. Computing bounded-width tree and branch decompositions of k-outerplanar graphs, 2013.

[24] D. Lapoire. Recognizability equals monadic second-order definability for sets of graphs of bounded tree-width. In *Proceedings STACS 1998*, volume 1373 of *LNCS*, pages 618–628. Springer, 1998.

[25] J. R. Myhill. Finite automata and the representation of events. Technical Report WADC TR-57-624, Wright-Paterson Air Force Base, 1957.

[26] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

[27] N. Robertson and P. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[28] M. M. Sysło. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26(1):47 – 53, 1979.

[29] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages. Beyond Words*, volume 3, pages 389–455. Springer, 1996.

[30] W. T. Tutte. *Connectivity in Graphs*. University of Toronto Press, 1966.

[31] W. T. Tutte. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1984.

[32] R. van Bevern, R. G. Downey, M. R. Fellows, S. Gaspers, and F. A. Rosamond. Myhill–Nerode methods for hypergraphs. *Algorithmica*, pages 1–34, 2015. in press.

[33] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[34] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–168, 1932.

[35] T. V. Wimer. *Linear Algorithms on K-terminal Graphs*. PhD thesis, Clemson University, Clemson, SC, USA, 1987.

A

Monadic Second Order Predicates and Sentences

We build sentences in monadic second order logic from a collection of predicates. Once we defined these predicates they will be the building blocks of more complex expressions, joined by MSOL-connectives and/or quantification of its declared variables. Hence, we follow the ideas of the work of Borie et al. [7], who also give a large list of predicates and their definitions.
Note that the length of our sentences and formulas always has to be bounded by some constant, independent of the size of the input graph.

We will denote single element variables by small letters, where $v, w, v', w', \ldots$ typically represent vertices and $e, f, e', f', \ldots$ edges. Set variables will be denoted by capital letters. Unless stated otherwise explicitly, V always denotes the vertex set of some input graph G and E its edge set. Since we always assume our predicates to appear in the context of such a graph we might drop these two variables as an argument of a predicate.

By some trivial definition, the following predicates are MSOL-definable (see also Theorem 1 in [7]). In our text we might refer to them as the *atomic* predicates of monadic second order logic over graphs.

(I) $v = w$ (Vertex equality)

(II) $\mathrm{Inc}(e, v)$ (Vertex-edge incidence)

(III) $v \in V$ (Vertex membership)

(IV) $e \in E$ (Edge membership)

Note that to shorten our notation we might omit statements such as $v \in V$ or $e \in E$ when quantifying over a variable. In this case we are referring to some vertex/edge in

the whole graph and the interpretation of the variables will always be obvious from the context or the notational conventions explained above.

From the atomic predicates, one can directly derive the following:

- $\mathrm{Adj}(v, w, E)$ (Adjacency of v and w in E)
- $\mathrm{Edge}(e, v, w)$ ($e = \{v, w\}$)

In a straightforward way (and by Theorem 4 in [7]), one can see that the following are MSOL-definable:

- $V = V' \cup V''$, $V = V' \setminus V''$, $V = V' \cap V''$ (plus the edge set equivalents)
- $V' = \mathrm{IncV}(E')$ [$E' = \mathrm{IncE}(V')$] (V' [E'] is the set of incident vertices [edges] of E' [V'])
- $\deg(v, E) = k$ (v has degree k in E, where k is a constant)
- $\mathrm{Conn}(V, E)$, $\mathrm{Conn}_k(V, E)$, $\mathrm{Cycle}(V, E)$, $\mathrm{Tree}(V, E)$, $\mathrm{Path}(V, E)$
- Minor_H (A graph contains a minor H of fixed size)

A.1 Edge Orientation of a Halin Graph

In the current section we show how to define an edge orientation on a Halin graph as explained in the proof of Proposition 4.2. That is, we will define a partition of the edge set of the graph into a directed tree E_T and a directed cycle E_C.

As outlined in the proof, we use a coloring on its vertex set to define the orientation of edges. Since we will use this result in later sections as well, we define the general case of a k-coloring on the vertices of a graph.

$$\mathrm{Part}_V(V, X_1, \ldots, X_k) \Leftrightarrow (\forall v \in V)\Big(\bigvee_{1 \le i \le k} v \in X_i \wedge \bigwedge_{\substack{1 \le i \le k \\ j \ne i}} \neg v \in X_j \Big)$$

$$\begin{aligned} k\text{-col}(X_1, \ldots, X_k) \Leftrightarrow & \mathrm{Part}_V(V, X_1, \ldots, X_k) \\ & \wedge \forall e \forall v \forall w \Big(\mathrm{Edge}(e, v, w) \rightarrow \bigwedge_{1 \le i \le k} \neg (v \in X_i \wedge w \in X_i) \Big) \end{aligned}$$

Now we define a predicate $\mathrm{head}(e, v)$ that is true if and only if v is the head vertex of the edge e in the given orientation by comparing the indices of the color classes that

contain an endpoint of e. Note that the following predicates always appear in the scope of an edge set F and a k-coloring $X_1, \ldots, X_k$.

$$\begin{aligned}
\mathrm{col}_<(v,w) \Leftrightarrow & \bigvee_{1\le i<j\le k} (v \in X_i \wedge w \in X_j) \\
\mathrm{head}(e,v) \Leftrightarrow & \exists w(\mathrm{Edge}(e,v,w) \wedge e \in F \leftrightarrow \mathrm{col}_<(v,w)) \\
\mathrm{tail}(e,v) \Leftrightarrow & \exists w(\mathrm{Edge}(e,v,w) \wedge \neg e \in F \leftrightarrow \mathrm{col}_<(v,w)) \\
\mathrm{Arc}(e,v,w) \Leftrightarrow & \mathrm{Edge}(e,v,w) \wedge \mathrm{head}(e,v) \quad [e=(v,w)]
\end{aligned}$$

Analogously to the definition of vertex degree predicates $\deg(v,E)$, as shown in [7, Theorem 4], we can define predicates $\deg_{\leftarrow}(v,E)$ and $\deg_{\rightarrow}(v,E)$ for the in-degree and out-degree of a vertex in a directed graph. We show how to define that the in-degree of a vertex is equal to a certain constant k.

$$\begin{aligned}
\deg_{\leftarrow}(v,E) \ge k \Leftrightarrow & \exists w_1 \cdots \exists w_k \Big(\Big(\bigwedge_{1\le i\le k} (\exists e \in E)\mathrm{Arc}(e,w_i,v) \Big) \\
& \wedge \bigwedge_{1\le i<j\le k} \neg w_i = w_j \Big) \\
\deg_{\leftarrow}(v,E) \le k \Leftrightarrow & \forall w_1 \cdots \forall w_{k+1} \Big(\Big(\bigwedge_{1\le i\le k+1} (\exists e \in E)\mathrm{Arc}(e,w_i,v) \Big) \\
& \rightarrow \bigvee_{1\le i<j\le k+1} w_i = w_j \Big) \\
\deg_{\leftarrow}(v,E) = k \Leftrightarrow & \deg_{\leftarrow}(v,E) \le k \wedge \deg_{\leftarrow}(v,E) \ge k
\end{aligned}$$

In a similar way we can define predicates for the out-degree and regularity of a vertex for in- and out-degree and both (denoted by k-reg$_{\leftarrow}$, k-reg$_{\rightarrow}$ and k-reg$_{\leftrightarrow}$, respectively). This enables us to define predicates for directed trees and cycles.

$$\begin{aligned}
\mathrm{Cycle}_{\rightarrow}(V,E) \Leftrightarrow & \mathrm{Conn}(V,E) \wedge 1\text{-reg}_{\leftrightarrow}(V,E) \\
\mathrm{Tree}_{\rightarrow}(V,E) \Leftrightarrow & \mathrm{Tree}(V,E) \wedge (\exists r \in V)(\forall v \in V) \Big((r = v \wedge \deg_{\leftarrow}(v,E) = 0) \\
& \vee (\neg v = r \wedge \deg_{\leftarrow}(v,E) = 1) \Big)
\end{aligned}$$

A.2 Child Ordering of a Halin Graph

This section concludes the proof of Lemma 4.3, that is we define an ordering on edges in a Halin graph that have the same parent in the tree E_T. Therefor we define predicates for directed paths and fundamental cycles. Note that $\mathrm{Path}_{\rightarrow}(s,t,E')$ is true if and only

if E' is a directed $s-t$-path.

$$\begin{aligned}\mathrm{Path}_{\rightarrow}(V,E) \Leftrightarrow &\mathrm{Tree}_{\rightarrow}(V,E) \wedge (\forall v \in V)\deg(v,E) \leq 2\\ \mathrm{Path}_{\rightarrow}(s,t,E') \Leftrightarrow &\mathrm{Path}_{\rightarrow}(\mathrm{IncV}(E'),E') \wedge \deg_{\leftarrow}(s) = 0 \wedge \deg_{\rightarrow}(t) = 0\end{aligned}$$

Now we turn to the notion of fundamental cycles. We assume that the following predicates appear within the scope of an edge set E_T, which is a spanning tree of the given graph.

$$\begin{aligned}\mathrm{FundCyc}(E') \Leftrightarrow &\mathrm{Cycle}(\mathrm{IncV}(E'),E') \wedge (\exists e \in E')(\forall e' \in E')(\neg(e = e') \leftrightarrow e \in E_T)\\ \mathrm{FundCyc}(e,e') \Leftrightarrow &(\exists E' \subseteq E)(e \in E' \wedge e' \in E' \wedge \mathrm{FundCyc}(E'))\end{aligned}$$

Note that $\mathrm{FundCyc}(e,e')$ is true if and only if there exists a fundamental cycle in the graph containing both e and e'. Now we can define an ordering $\mathrm{nb}_{<}(e,f)$ on edges with the same parent, as explained in the proof of Lemma 4.3.

$$\begin{aligned}\mathrm{nb}_{<}(e,f) \Leftrightarrow &\mathrm{head}(e) = \mathrm{head}(f) \wedge (\exists f' \in E_C)(\forall e' \in E_C)(\forall F' \subseteq E_C)(\forall E' \subseteq E_C)\\ &\Big(\Big(\mathrm{FundCyc}(e,e') \wedge \mathrm{FundCyc}(f,f') \wedge \mathrm{Path}_{\rightarrow}(r,\mathrm{tail}(e'),E')\\ &\quad \wedge \mathrm{Path}_{\rightarrow}(r,\mathrm{tail}(f'),F')\Big) \rightarrow F' \subset E'\Big)\end{aligned}$$

Furthermore we define a predicate $\mathrm{nb}_{\prec}(e,f)$ that is true if and only if f is the leftmost right neighbor of e and vice versa. We also apply this notion to vertex variables, which allows us to refer to left and right siblings of a vertex. We denote these predicates by $\mathrm{sib}_{<}(x,y)$ and $\mathrm{sib}_{\prec}(x,y)$.

$$\begin{aligned}\mathrm{nb}_{\prec}(e,f) \Leftrightarrow &\mathrm{nb}_{<}(e,f) \wedge \forall f'((\neg f = f' \wedge \mathrm{nb}_{<}(e,f')) \rightarrow \mathrm{nb}_{<}(f,f'))\\ \mathrm{sib}_{<}(x,y) \Leftrightarrow &\exists e \exists f(\mathrm{tail}(e,x) \wedge \mathrm{tail}(f,y) \wedge \mathrm{nb}_{<}(e,f))\\ \mathrm{sib}_{\prec}(x,y) \Leftrightarrow &\exists e \exists f(\mathrm{tail}(e,x) \wedge \mathrm{tail}(f,y) \wedge \mathrm{nb}_{\prec}(e,f))\end{aligned}$$

In the following we will use the rewrite of $\mathrm{sib}_{\prec}$ to

$$y = l(x) \Leftrightarrow \mathrm{sib}_{\prec}(y,x).$$

This expresses that a vertex y is the direct left sibling of the vertex x in our ordering.

A.3 Tree Decomposition of a Halin Graph

In this section we define predicates $\mathrm{Bag}_{\sigma}(e,X)$ for all bag types used in the proof of Lemma 4.5, and $\mathrm{Parent}(X_p,X_c)$ according to the given construction. In the following we assume that we are given an edge $e \in E_T$, $e = \{x,y\}$, such that y is the parent of x in E_T.

A.3.1 Boundary vertices

For defining predicates for bag types in our tree decomposition, we need to show how to define boundary vertices in MSOL. First, we define predicates to check whether a vertex is the right-(/left-)most child of its parent.

$$\text{Child}_{R+}(x) \Leftrightarrow \forall y \forall z \forall e \forall e' ((\text{Arc}(e, y, x) \wedge \text{Arc}(e', y, z)) \rightarrow \text{nb}_{<}(e', e))$$

Note that $\text{Child}_{L+}(x)$ can be defined similarly, replacing $\text{nb}_{<}(e', e)$ by $\text{nb}_{<}(e, e')$. In the following we let $V_C = \text{IncV}(E_C)$.

$$\begin{aligned} y = bd_r(x) \Leftrightarrow & (x \in V_C \wedge x = y) \vee \Big(x \in V \wedge y \in V_C \\ & \wedge \Big((\exists E_P \subseteq E_T)(\text{Path}_{\rightarrow}(x, y, E_P) \wedge (\forall e \in E_P) \\ & (\forall z(\text{tail}(e, z) \rightarrow \text{Child}_{R+}(z)))) \Big) \Big) \end{aligned}$$

Replaying Child_{R+} by Child_{L+} in the above predicate we can also define $y = bd_l(x)$.

A.3.2 Bag Types

We define an MSOL-predicate for each bag type that we introduced in the proof of Lemma 4.5. Using the definition of boundary vertices given above, we can define them in a straightforward manner.

$$\begin{aligned} \text{Bag}_{R1}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = x \vee x' = bd_r(x) \vee x' = bd_l(x)) \\ \text{Bag}_{R2}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = x \vee x' = bd_r(x) \vee x' = bd_l(x)) \\ \text{Bag}_{R3}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = bd_r(x) \vee x' = bd_l(x)) \\ \text{Bag}_{L1}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = bd_l(y) \vee bd_r(l(x))) \\ \text{Bag}_{L2}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = bd_l(y) \vee x' = bd_r(l(x)) \vee x' = bd_l(x)) \\ \text{Bag}_{L3}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = bd_l(y) \vee x' = bd_l(x)) \\ \text{Bag}_{LR}(e, X) \Leftrightarrow & (x' \in X) \leftrightarrow (x' = y \vee x' = bd_l(y) \vee x' = bd_r(x) \vee x' = bd_l(x)) \end{aligned}$$

As a next step we will unify the above predicates, to deal with the cases when certain bags do not need to be created for an edge. This is the case when we reach the root vertex of the graph or whenever an edge is the leftmost child edge of a vertex.

$$\begin{aligned} \text{Bag}(X) \Leftrightarrow & \exists e \Big(y = r \wedge (\text{Bag}_{R1}(e, X) \vee \text{Bag}_{R2}(e, X)) \\ & \vee \Big(\neg y = r \wedge \Big((\text{Child}_{L+}(x) \wedge (\text{Bag}_{R1}(e, X) \vee \text{Bag}_{R2}(e, X) \\ & \vee \text{Bag}_{R3}(e, X))) \vee (\neg \text{Child}_{L+}(x) \wedge (\text{Bag}_{R1}(e, X) \\ & \vee \cdots \vee \text{Bag}_{LR}(e, X))) \Big) \Big) \Big) \end{aligned}$$

A.3.3 The Parent Relation

We now turn to defining the predicate $\text{Parent}(X_p, X_c)$, which is true if and only if the bag X_p is the parent bag of X_c in the tree decomposition. Due to the contraction step we can only have edges between bags if their vertex sets are not equal. Note that adding the term '$\neg X_p = X_c$' is sufficient to represent these contractions. The rest is a case analysis as implied by Figure 4.2b and the respective parent/child relationships between components.

$$\begin{aligned}
\text{Parent}(X_p, X_c) \Leftrightarrow & \text{Bag}(X_p) \wedge \text{Bag}(X_c) \wedge \neg X_p = X_c \wedge (\text{Parent}_I(X_p, X_c) \\
& \vee \text{Parent}_{NB}(X_p, X_c) \vee \text{Parent}_P(X_p, X_c)) \\
\text{Parent}_I(X_p, X_c) \Leftrightarrow & \exists e \Big((\text{Bag}_{R1}(e, X_c) \wedge \text{Bag}_{R2}(e, X_p)) \\
& \vee (\text{Bag}_{R2}(e, X_c) \wedge \text{Bag}_{R3}(e, X_p)) \\
& \vee ((\text{Bag}_{R3}(e, X_c) \vee \text{Bag}_{L3}(e, X_c)) \wedge \text{Bag}_{LR}(e, X_p)) \\
& \vee (\text{Bag}_{L1}(e, X_c) \wedge \text{Bag}_{L2}(e, X_p)) \\
& \vee (\text{Bag}_{L2}(e, X_c) \wedge \text{Bag}_{L3}(e, X_p)) \Big) \\
\text{Parent}_{NB}(X_p, X_c) \Leftrightarrow & \exists e \exists e' (\text{nb}_{\prec}(e, e') \wedge \text{Bag}_{LR}(e, X_c) \wedge \text{Bag}_{L1}(e', X_p)) \\
\text{Parent}_P(X_p, X_c) \Leftrightarrow & \exists e \exists e' (\text{Child}_{R+}(x) \wedge \text{tail}(e', y) \\
& \wedge \text{Bag}_{LR}(e, X_c) \wedge \text{Bag}_{R1}(e', X_p))
\end{aligned}$$

A.4 Equivalence Class Membership for Halin Graphs

In this section we complete the proof of Lemma 4.8, which states that finite index implies MSOL-definability for Halin graphs. In particular we define the predicates ϕ_{Leaf}, ϕ_{TSG} and ϕ_{Root}, which represent the cases for leaf bags, inner bags (i.e., intermediate and branch bags that are not the root) and the root bag, respectively.

The predicate ϕ_{Leaf} can be defined in a straightforward way, using the fact that we know that all terminal subgraphs of leaf bags are in the equivalence class C_{Leaf} and that leaf bags are always of type $R1$.

$$\phi_{Leaf} = \forall X \forall e((\text{Bag}_{R1}(e, X) \wedge \text{Leaf}(X)) \rightarrow e \in C_{Leaf,R1})$$

Next, we turn to defining ϕ_{TSG}, where we distinguish two cases. That is, either X is an intermediate or a branch bag. We conduct the case analysis as implied by the

construction of our tree decomposition as shown in Section 4.2.

$$\begin{aligned}\phi_{TSG} =&\Big(\exists C_{i,L1}\exists C_{i,L2}\exists C_{i,L3}\exists C_{i,R1}\exists C_{i,R2}\exists C_{i,R3}\exists C_{i,LR}\Big)_{i=1,\dots,r}\\ &\forall X\forall Y\Big((\mathrm{Parent}(X,Y)\wedge\mathrm{Int}(X))\rightarrow\phi_{TSG,Int}\\ &\wedge\forall Y'(\neg(Y=Y')\wedge\mathrm{Parent}(X,Y)\wedge\mathrm{Parent}(X,Y')\wedge\mathrm{Branch}(X))\\ &\rightarrow\phi_{TSG,Branch}\Big)\end{aligned}$$

The first case we are considering is when X is an intermediate node with child bag Y. These edges either belong to the same component, which is handled in the first part of the predicate, or they belong to components of different edges, such that the two are either direct neighbor edges according to the $\mathrm{nb}_{\prec}$-ordering or one of the edges is the parent edge of the other one.

$$\begin{aligned}\phi_{TSG,Int} =&\forall e\Big((\mathrm{Bag}_{L2}(e,X)\wedge\mathrm{Bag}_{L1}(e,Y))\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,L1}\rightarrow e\in C_{f_I(i,X),L2})\\ &\vee(\mathrm{Bag}_{L3}(e,X)\wedge\mathrm{Bag}_{L2}(e,Y))\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,L2}\rightarrow e\in C_{f_I(i,X),L3})\\ &\vee(\mathrm{Bag}_{R2}(e,X)\wedge\mathrm{Bag}_{R1}(e,Y))\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,R1}\rightarrow e\in C_{f_I(i,X),R2})\\ &\vee(\mathrm{Bag}_{R3}(e,X)\wedge\mathrm{Bag}_{R2}(e,Y))\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,R2}\rightarrow e\in C_{f_I(i,X),R3})\Big)\\ &\vee\forall e\forall e'\Big((\mathrm{Parent}_{NB}(X,Y)\wedge\mathrm{Bag}_{L1}(e',X)\wedge\mathrm{Bag}_{LR}(e,Y))\\ &\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,LR}\rightarrow e'\in C_{f_I(i,X),L1})\Big)\\ &\vee\Big((\mathrm{Parent}_{P}(X,Y)\wedge\mathrm{Bag}_{R1}(e',X)\wedge\mathrm{Bag}_{LR}(e,Y))\\ &\rightarrow\bigwedge_{i=1,\dots,r}(e\in C_{i,LR}\rightarrow e'\in C_{f_I(i,X),R1})\Big)\end{aligned}$$

Now we assume that X is a branch node with child bags Y and Y'. We can't identify the types of the bags Y and Y' immediately, since some of the edges in the component might have been contracted. So in the following, let L denote the type $L1, L2$ or $L3$, and R, respectively, $R1, R2$ or $R3$. We can define each combination of the actual types in exactly the same way.

$$\begin{aligned}\phi_{TSG,Branch} =&\forall e\Big((\mathrm{Bag}_{LR}(e,X)\wedge\mathrm{Bag}_{L}(e,Y)\wedge\mathrm{Bag}_{R}(e,Y'))\\ &\rightarrow\bigwedge_{\substack{i=1,\dots,r\\ j=1,\dots,r}}((e\in C_{i,L}\wedge e\in C_{j,R})\rightarrow e\in C_{f_J(\{i,j\},X),LR})\Big)\end{aligned}$$

Knowing that all graphs that have property P are contained in one of the equivalence classes $C_{A_1},\dots,C_{A_p}$ and that the root bag is always of type $R2$, we can define ϕ_{Root}

directly.

$$\phi_{Root} = \forall X \forall e \Big((\mathrm{Root}(X) \wedge \mathrm{Bag}_{R2}(e, X)) \rightarrow \bigvee_{i=A_1,\ldots,A_p} e \in C_{i,R2} \Big)$$

A.5 Equivalence Class Membership - Generalized

In the current section we describe how to define predicates for the equivalence class membership of (partial) terminal subgraphs in any MSOL-definable ordered tree decomposition, hence concluding the proof of Lemma 5.1. In this case we do not know the specific shape of the tree decomposition, so our case analysis becomes somewhat more lengthy. We give examples for each predicate involved from which it will become apparent that one can define any such case in a similar way.
Once we defined all predicates for MSOL-definable ordered tree decompositions, we additionally show how to define the case of branch nodes in an MSOL-definable tree decomposition of bounded degree, hence concluding the proof of Lemma 5.3.

As before (Appendix A.4) we first define all sets that we need for the predicates and then distinguish the cases that X is an intermediate node or a branch node. These predicates will be defined in detail in the following sections.

$$\begin{aligned}\phi_{TSG} =& \Big(\exists C^V_{i,\tau} \exists C^E_{i,\sigma} \exists C^{V|P}_{i,\tau} \exists C^{E|P}_{i,\sigma} \exists C^{V|C}_{i,\tau} \exists C^{E|C}_{i,\sigma} \Big)_{\substack{i=1,\ldots,r \\ \sigma,\in\{\sigma_1,\ldots,\sigma_s\} \\ \tau,\in\{\tau_1,\ldots,\tau_t\}}} \\ & \Big(\phi_{TSG,Int} \wedge \phi_{TSG,Branch} \Big)\end{aligned}$$

A.5.1 Intermediate Nodes

First, we define the equivalence class membership for terminal subgraphs corresponding to an intermediate node in the tree decomposition. We conduct a case analysis as discussed in the proof of Lemma 5.1 w.r.t. the types of the bags X and Y.

$$\begin{aligned}\phi_{TSG,Int} =& \forall X \forall Y \Big((\mathrm{Int}(X) \wedge \mathrm{Parent}(X, Y)) \\ & \rightarrow \bigwedge_{\substack{\tau,\tau' \in \{\tau_1,\ldots,\tau_t\} \\ \sigma,\sigma' \in \{\sigma_1,\ldots,\sigma_s\}}} \Big(\phi_{Int,\tau,\tau'} \wedge \phi_{Int,\sigma,\sigma'} \wedge \phi_{Int,\tau,\sigma} \wedge \phi_{Int,\sigma,\tau} \Big) \Big) \end{aligned} \tag{A.1}$$

Case (I) Both bags belong to a vertex. For each pair of types $\tau, \tau' \in \{\tau_1, \ldots, \tau_t\}$ one can define the following predicate.

$$\begin{aligned}\phi_{Int,\tau,\tau'} =& \forall v \forall v' \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^V_{\tau'}(v', Y)) \\ & \rightarrow \bigwedge_{i=1,\ldots,r} (v' \in C^V_{i,\tau'} \rightarrow v \in C^V_{f_I(i,X),\tau}) \Big)\end{aligned}$$

Case (II) Both bags belong to an edge. For each pair of types $\sigma, \sigma' \in \{\sigma_1, \ldots, \sigma_s\}$ we can write down a similar predicate.

$$\begin{aligned}\phi_{Int,\sigma,\sigma'} =& \forall e \forall e' \Big((\mathrm{Bag}^E_\sigma(e, X) \wedge \mathrm{Bag}^E_{\sigma'}(e', Y)) \\ & \rightarrow \bigwedge_{i=1,\ldots,r} (e' \in C^E_{i,\sigma'} \rightarrow e \in C^E_{f_I(i,X),\sigma}) \Big)\end{aligned}$$

Case (III) The bag X belongs to a vertex and Y belongs to an edge. For each pair of a type $\tau \in \{\tau_1, \ldots, \tau_t\}$ and $\sigma \in \{\sigma_1, \ldots, \sigma_s\}$ one can define:

$$\begin{aligned}\phi_{Int,\tau,\sigma} =& \forall v \forall e \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^E_\sigma(e, Y)) \\ & \rightarrow \bigwedge_{i=1,\ldots,r} (e \in C^E_{i,\sigma} \rightarrow v \in C^V_{f_I(i,X),\tau}) \Big)\end{aligned}$$

Case (IV) The bag X belongs to an edge and Y belongs to a vertex. For σ, τ as above we define:

$$\begin{aligned}\phi_{Int,\sigma,\tau} =& \forall e \forall v \Big((\mathrm{Bag}^E_\sigma(e, X) \wedge \mathrm{Bag}^V_\tau(v, Y)) \\ & \rightarrow \bigwedge_{i=1,\ldots,r} (v \in C^V_{i,\tau} \rightarrow e \in C^E_{f_I(i,X),\sigma}) \Big)\end{aligned}$$

A.5.2 Branch Nodes

In the following we will define predicates for branch nodes in an ordered MSOL-definable tree decomposition, such that all bags considered always correspond to vertices in the graph. Note that in the cases that some of them are edge bags, one can write down all predicates in the same way (replacing some vertices/vertex sets with edges/edge sets in the predicates).
First we define the general case, in which Y is neither the leftmost nor the rightmost child of X and deal with the special cases later. Let Y' is the direct right sibling of Y.

$$\begin{aligned}\phi^I_{Branch,\tau,\tau',\tau''} =& \forall v \forall v' \forall v'' \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^V_{\tau'}(v', Y) \wedge \mathrm{Bag}^V_{\tau''}(v'', Y')) \\ & \rightarrow \bigwedge_{i=1,\ldots,r} \Big(\Big(v \in C^{V|P}_{i,\tau} \wedge v' \in C^{V|C}_{i,\tau'} \wedge v' \in C^V_{j,\tau'} \Big) \\ & \rightarrow \Big(v \in C^{V|P}_{f_J(i,j),\tau} \wedge v'' \in C^{V|C}_{f_J(i,j),\tau''} \Big) \Big) \Big)\end{aligned}$$

Now we consider the situation when Y is the leftmost child of X with right sibling Y'. In this case we derive the partial terminal subgraph $[X]^+_{Y'}$ by pretending that Y is the only child of X and using the method for intermediate nodes. It is easy to see that this

way we indeed define the equivalence class membership for $[X]^+_{Y'}$.

$$\phi^{L+}_{Branch,\tau,\tau',\tau''} = \forall v \forall v' \forall v'' \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^V_{\tau'}(v', Y) \wedge \mathrm{Bag}_{\tau''}(v'', Y')) \\ \rightarrow \bigwedge_{i=1,\dots,r} \Big(v' \in C^V_{i,\tau'} \rightarrow \Big(v \in C^{V|P}_{f_I(i,X),\tau} \wedge v'' \in C^{V|C}_{f_I(i,X),\tau''} \Big) \Big) \Big)$$

When reaching the rightmost child of a branch bag X, we derive the terminal subgraph $[X]^+$. Assume in the following that Y is the rightmost child of X.

$$\phi^{R+}_{Branch,\tau,\tau'} = \forall v \forall v' \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^V_{\tau'}(v', Y)) \\ \rightarrow \bigwedge_{i=1,\dots,r} \Big(\Big(v \in C^{V|P}_{i,\tau} \wedge v' \in C^{V|C}_{i,\tau'} \wedge v' \in C^V_{j,\tau'} \Big) \rightarrow v \in C^V_{f_J(i,j),\tau} \Big) \Big)$$

One can define a predicate $\phi_{TSG,Branch}$ in a similar way as $\phi_{TSG,Int}$ using the predicates described above together with $\mathrm{Child}_{L+}(X)$, $\mathrm{Child}_{R+}(X)$ and $\mathrm{nb}_\prec(X, Y)$. Disregarding the types of bags for now, one can define the predicate $\phi'_{TSG,Branch}$ in the following way.

$$\phi'_{TSG,Branch} = \forall X \forall Y \Big((\mathrm{Parent}(X, Y) \wedge \mathrm{Branch}(X)) \rightarrow \Big(\Big(\mathrm{Child}_{R+}(Y) \wedge \phi^{R+}_{JoinB} \Big) \\ \vee \forall Y' \Big(\mathrm{nb}_\prec(Y, Y') \rightarrow \Big(\Big(\mathrm{Child}_{L+} \wedge \phi^{L+}_{\mathrm{Branch}} \Big) \\ \vee \Big(\neg \mathrm{Child}_{L+}(Y) \wedge \phi^{I}_{\mathrm{Branch}} \Big) \Big) \Big) \Big) \Big)$$

Note that to include the case analysis, one can define a predicate $\phi_{TSG,Branch}$ as it is done in the definition of $\phi_{TSG,Int}$ (Predicate A.1), for all combinations of vertex/edge types.

A.5.3 Branch Nodes for Bounded Degree Tree Decompositions

To finish the proof of Lemma 5.3, we only have to show how to define a predicate for branch nodes with a constant number of children as explained in the proof.
Again, we give an example predicate for the case that all bags involved are vertex bags and note that all other cases can be defined similarly. Consider a branch bag X with child bags $X_1, \dots, X_k$, all corresponding to vertices in the graph and types $\tau_1, \dots, \tau_k$. Then we can define this predicate as follows.

$$\phi_{Branch,\tau,\tau_1,\dots,\tau_k} = \forall v \forall v_1 \cdots \forall v_k \Big((\mathrm{Bag}^V_\tau(v, X) \wedge \mathrm{Bag}^V_{\tau_1}(v_1, X_1) \\ \wedge \cdots \wedge \mathrm{Bag}^V_{\tau_k}(v_k, X_k)) \rightarrow \bigwedge_{\substack{i_k=1,\dots,r \\ \cdots \\ i_1=1,\dots,r}} \Big((v_1 \in C^V_{i_1,\tau_1} \\ \wedge \cdots \wedge v_k \in C^V_{i_k,\tau_k}) \rightarrow v \in C^V_{f_J(\{i_1,\dots,i_k\},X),\tau} \Big) \Big)$$

A.5.4 Counting for Branch Nodes of Unbounded Degree

In the following, assume that X is a branch bag with an unbounded number of children. The first step in defining the equivalence class membership of the terminal subgraph w.r.t. X in CMSOL is to define a predicate to check whether a group i w.r.t. an equivalence class C_i has a certain size modulo $r!$. Since we do not know whether its children are associated with vertices or edges in the graph, we have to define a number of sets $X^V_{i,\tau}$ and $X^E_{i,\sigma}$, containing vertices (edges) such that $X^V_{i,\tau}$ contains a vertex v (edge e) if and only if X has a child of type τ (σ) whose terminal subgraph is in equivalence class C_i.

$$V' = X^V_{i,\tau} \Leftrightarrow v \in V' \leftrightarrow \exists Y(\text{Parent}(X,Y) \wedge \text{Bag}^V_\tau(v,Y) \wedge v \in C^V_{i,\tau})$$
$$E' = X^E_{i,\sigma} \Leftrightarrow e \in E' \leftrightarrow \exists Y(\text{Parent}(X,Y) \wedge \text{Bag}^E_\sigma(e,Y) \wedge e \in C^E_{i,\sigma})$$

Next, we conduct a case analysis to see whether the sizes of these sets added together indeed have size c (modulo $r!$), which implies that the number of children modulo $r!$ is equal to c.

$$\begin{aligned}|\text{Group}^X_i| \bmod r! = c \Leftrightarrow &\Big(\exists X^V_{i,\tau} \exists X^E_{i,\sigma}\Big)_{\substack{\tau\in\{\tau_1,\ldots,\tau_t\}\\ \sigma\in\{\sigma_1,\ldots,\sigma_s\}}} \\ &\bigvee_{\substack{(\sum_\tau c_\tau + \sum_\sigma c_\sigma) \bmod r! = c \\ \sum_\tau c_\tau + \sum_\sigma c_\sigma \leq r!\cdot(s+t)}} \Big(|X^V_{i,\tau_1}| \bmod r! = c_{\tau_1} \wedge \cdots \wedge |X^V_{i,\tau_t}| \bmod r! = c_{\tau_t} \\ &\qquad \wedge |X^E_{i,\sigma_1}| \bmod r! = c_{\sigma_1} \wedge \cdots \wedge |X^E_{i,\sigma_s}| \bmod r! = c_{\sigma_s}\Big)\end{aligned}$$

To define the equivalence class membership of partial terminal group subgraphs, we first define the corresponding sets (and assume that all sets for terminal subgraphs are already defined as before).

$$\Big(\exists C^{\mathcal{G}_i|V}_{j,\tau} \exists C^{\mathcal{G}_i|E}_{j,\sigma}\Big)_{\substack{i=1,\ldots,r\\ j=\epsilon,1,\ldots,r\\ \tau\in\{\tau_1,\ldots,\tau_t\}\\ \sigma\in\{\sigma_1,\ldots,\sigma_s\}}}$$

In the following we assume that X is associated with vertex v and type τ and note that we can define the edge cases analogously. We first consider the case, when the group i w.r.t. an equivalence class C_i is empty.

$$\bigwedge_{i=1,\ldots,r} \Big(\Big(\bigwedge_{\tau'} X^V_{i,\tau'} = \emptyset \wedge \bigwedge_\sigma X^E_{i,\sigma} = \emptyset\Big) \rightarrow v \in C^{\mathcal{G}_i|V}_{\epsilon,\tau}\Big)$$

For all i, such that $v \notin C^{\mathcal{G}_i|V}_{\epsilon,\tau}$, we define the equivalence class membership of the partial terminal group subgraph of X w.r.t. C_i using the arguments given in the proof of Lemma

5.6. Let $j = f_I(i, X)$.

$$\bigwedge_{c=1,\dots,r!} |\text{Group}_i^X| \bmod r! = c \to v \in C^{\mathcal{G}_i|V}_{f^c_j(j),\tau}$$

Since now we are left with a bounded number of r cases, we can use the function $f_{\mathcal{G}}$ to determine the equivalence class membership of $[X]^+$.

$$\bigwedge_{\substack{j_1=\epsilon,1,\dots,r \\ \dots \\ j_r=\epsilon,1,\dots,r}} \Big(\Big(v \in C^{\mathcal{G}_1|V}_{j_1,\tau} \wedge \dots \wedge v \in C^{\mathcal{G}_r|V}_{j_r,\tau}\Big) \to v \in C^V_{f_{\mathcal{G}}(j_1,\dots,j_r),\tau}\Big)$$

This completes our predicate.

A.6 k-Cycle Trees

In the current section we give all predicates to define a tree decomposition of a k-cycle tree in MSOL, as explained in the proof of Lemma 5.16. We first define the edge orientation ϕ_{Ori} and then all predicates for the bag types. Note that since this construction is very similar to the one for Halin graphs, we do not define the Parent-predicate explicitly, as it works in almost the exact same way.

As a first step we define a predicate to check whether two vertices have a certain (constant) distance in a given edge set.

$$\text{dist}(v, w, E') = k \Leftrightarrow (\exists E_P \subseteq E')(\text{Path}(v, w, E_P) \wedge |E_P| = k)$$

This allows us to define the the i-th cycle of the graph.

$$E' = \text{Cycle}_i \Leftrightarrow \text{Cycle}_{\to}(\text{IncV}(E'), E') \wedge \forall v(\text{Inc}(v, E') \to \text{dist}(c, v, E) = i)$$

We can write down the orientation ϕ_{Ori} described in the proof of Lemma 5.16 in the following way.

$$\begin{aligned}\phi_{Ori} =& \exists E_T \exists E_{C_1} \cdots \exists E_{C_k} \exists r_1 \cdots \exists r_{k-1} \Big((\text{Part}_E(E, E_T, E_{C_1}, \dots, E_{C_k}) \\ & \wedge \text{Tree}_{\to}(V, E_T) \wedge \bigwedge_{i=1,\dots,k} E_{C_i} = \text{Cycle}_i \\ & \wedge \bigwedge_{i=1,\dots,k-1} \Big(r_i \in \text{IncV}(E_{C_i}) \wedge \text{dist}(r, r_i, E_T) = k - i \Big) \Big)\end{aligned}$$

We can define a predicate $\text{nb}^i_<(e, f)$ in complete analogy to $\text{nb}_<(e, f)$ as shown in Appendix A.2 by simply replacing E_C by E_{C_i} and r by r_i (for the case that $i = k$ don't have to modify it). This predicate is true if and only if e is on the left of f, such that e and f have the same head vertex, i.e. their tail vertices lie on the same cycle.

Now we turn to defining the i-th boundary vertex (Definition 5.15).

$$\begin{aligned} w = bd_i^r(v, E') \Leftrightarrow & (w \in V_{C_i} \wedge v = w) \\ & \vee (\exists E_P \subseteq E')(\mathrm{Path}_{\rightarrow}(v, w, E_P) \wedge w \in \mathrm{IncV}(E_{C_i})) \\ & \wedge (\forall e \in E_P) \neg (\exists E'_P \subseteq E') \Big(\mathrm{Path}_{\rightarrow}(v, w, E'_P) \wedge (\exists e' \in E'_P) \\ & \bigvee_{i=1,\ldots,k} \mathrm{nb}^i_{<}(e, e') \Big) \end{aligned} \tag{A.2}$$

To define bd_i^l, we simply replace $\mathrm{nb}^i_{<}(e, e')$ by $\mathrm{nb}^i_{<}(e', e)$ in line A.2. In the following we abbreviate $w = bd^i(v, E_T)$ to $w = bd^i(v)$. We denote by $NB_R(e)$ the edge set containing e and all its right neighbor edges.

We are now equipped with all tools to define the bag types for a tree decomposition of a k-cycle tree. We use the same notation as in Appendix A.3, that is, we have an edge $e = \{x, y\}$, such that y is the parent of x in E_T and assume that the vertex y lies on cycle C_i. The predicate $\mathrm{CarryBD}^r$ defines the case that the vertex x does not have a left boundary on a cycle C_j, so that we have to pass on the right boundary vertex of y without the edge e and its right neighbors.

$$\mathrm{CarryBD}^r(e, z)_j \Leftrightarrow (\neg(\exists z'(z' = bd_j^l(x))) \wedge z = bd_j^r(y, E_T \setminus NB_R(e))$$

We continue by defining the bag types $R1, \ldots, LR$.

$$\mathrm{Bag}_{R1}(e, X) \Leftrightarrow z \in X \leftrightarrow \bigvee_{i<j\leq k} \Big(z = bd_j^l(x) \vee z = bd_j^r(x) \Big)$$

$$\mathrm{Bag}_{R2}(e, X) \Leftrightarrow z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i<j\leq k} \Big(z = bd_j^l(x) \vee z = bd_j^r(x) \Big) \Big)$$

$$\begin{aligned} \mathrm{Bag}_{L1}(e, X) \Leftrightarrow & z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i\leq j\leq k} \Big(z = bd_j^l(y) \\ & \vee z = bd_j^r(y, E_T \setminus NB_R(e)) \Big) \Big) \\ \mathrm{Bag}_{L2}(e, X) \Leftrightarrow & z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i<j\leq k} \Big(z = bd_j^l(y) \vee z = bd_j^l(x) \\ & \vee z = bd_j^r(y, E_T \setminus NB_R(e)) \Big) \Big) \end{aligned}$$

$$\mathrm{Bag}_{L3}(e,X) \Leftrightarrow z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i<j\leq k} \Big(z = bd^l_j(y) \vee z = bd^l_j(x) \vee \mathrm{CarryBD}^r(e,z)\Big)\Big)$$

$$\mathrm{Bag}_{LR}(e,X) \Leftrightarrow z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i<j\leq k} \Big(z = bd^l_j(x) \vee z = bd^r_j(x) \vee z = bd^l_j(y) \vee \mathrm{CarryBD}^r(e,z)\Big)\Big)$$

Note that defining the Parent-predicate works in the same way as for Halin graphs, taking into account the missing bag type $R3$.

A.7 Adding Feedback Edge/Vertex Sets

In this section we complete the proof of Theorem 5.19. In the following, let $G' = (V', E')$ and $G = (V, E)$ be graphs as stated in Theorem 5.19. Assume that we are given predicates Bag^V_τ and Bag^E_σ for vertex bag types τ and σ, defined for vertices and edges of the spanning tree E_T of a graph, defining a tree decomposition of G'. One can observe that we can define the sets V' and (a set representing) E' easily, using the following facts.

- Each vertex $v' \in V'$ is contained in a bag of the tree decomposition, i.e. (at least) one of the Bag-predicates evaluates to *true* for some set $X \subseteq V$.
- For each edge $e' \in E'$ there is a bag containing both endpoints. Note that if there is an edge in $E \setminus E'$, such that both its endpoints are contained in a bag, we do not need to consider it any further.

In the following we assume that V' and E' are defined and FundCyc uses the maximal spanning tree E_T, upon which the construction of the tree decomposition of G' is based. First, we consider the case of feedback edge sets. We use the notion of fundamental cycles rather that directly referring to biconnected components, since it makes our predicate shorter (while in this case they express the same thing).[(i)]

$$\mathrm{Bag}^{E,+}_\sigma(e,X) \Leftrightarrow v' \in X \leftrightarrow \Big((\exists e' \in E \setminus E')\Big(\mathrm{Inc}(v',e') \wedge \mathrm{FundCyc}(e,e') \wedge \forall w((\neg v' = w \wedge \mathrm{Inc}(w,e')) \rightarrow \mathrm{col}(v') < \mathrm{col}(w))\Big)\Big)$$

$$\mathrm{Bag}^{V,+}_\tau(v,X) \Leftrightarrow v' \in X \leftrightarrow \Big((\exists e' \in E \setminus E')\Big(\mathrm{Inc}(v',e') \wedge \mathrm{FundCyc}(v,e') \wedge \forall w((\neg v' = w \wedge \mathrm{Inc}(w,e')) \rightarrow \mathrm{col}(v') < \mathrm{col}(w))\Big)\Big)$$

[(i)] Note that the predicate FundCyc can easily be defined for a combination of a vertex and an edge as well.

For feedback vertex sets we can define similar additions to the respective predicates, directly using the biconnected components mentioned in the proof.

$$\begin{aligned}\mathrm{Bag}_\tau^{V,+}(v,X) \Leftrightarrow & v' \in X \leftrightarrow \Big((\exists V_2 \subseteq V)(v \in V_2 \wedge v' \in V_2 \\ & \wedge \mathrm{Conn}_2(V_2, E_T \cup \mathrm{IncE}(V_2 \setminus V')))\Big) \\ \mathrm{Bag}_\sigma^{E,+}(e,X) \Leftrightarrow & v' \in X \leftrightarrow \Big((\exists V_2 \subseteq V)\Big(v' \in V_2 \wedge e' \in \mathrm{IncE}(V_2 \setminus V') \\ & \wedge \mathrm{Conn}_2(V_2, E_T \cup \mathrm{IncE}(V_2 \setminus V'))\Big)\Big)\end{aligned}$$

A.8 Bounded Vertex and Edge Remember Number

As the last of our extensions, we show how to define tree decompositions that have a bounded vertex and edge remember number. Hence, we will conclude the proof of Theorem 6.2, which we used to prove the case for bounded degree k-outerplanar graphs.

First, we are going to show how to identify an edge set as a spanning tree with vertex remember number less than or equal to κ and edge remember number less than or equal to λ, both constant.

$$\begin{aligned}& \exists E_T(\mathrm{Tree}(V, E_T) \wedge vr(E_T) \leq \kappa \wedge er(E_T) \leq \lambda) \\ vr(E_T) \leq \kappa \Leftrightarrow & (\forall v \in V)(\forall e_1 \in E \setminus E_T) \cdots \forall (e_{\kappa+1} \in E \setminus E_T) \\ & \Big(\Big(\bigwedge_{i=1,\ldots,\kappa+1} \mathrm{FundCyc}(v, e_i)\Big) \rightarrow \bigvee_{1 \leq i < j \leq \kappa+1} e_i = e_j\Big) \\ er(E_T) \leq \lambda \Leftrightarrow & (\forall e \in E)(\forall e_1 \in E \setminus E_T) \cdots \forall (e_{\lambda+1} \in E \setminus E_T) \\ & \Big(\Big(\bigwedge_{i=1,\ldots,\lambda+1} \mathrm{FundCyc}(e, e_i)\Big) \rightarrow \bigvee_{1 \leq i < j \leq \lambda+1} e_i = e_j\Big)\end{aligned}$$

In the following, assume that E_T is the edge set of the spanning tree of G (as shown above), which additionally has edge orientations, defined in MSOL by predicates head and tail (cf. Appendix A.1). Note that the last predicate in the list, $\mathrm{nb}_<(X_a, X_b)$ requires an ordering on edges with the same head vertex.

$$\begin{aligned}\mathrm{Bag}_V(v,X) \Leftrightarrow & v' \in X \leftrightarrow (v' = v \vee (\exists e \in E \setminus E_T)(\mathrm{Inc}(v', e) \\ & \wedge \mathrm{FundCyc}(v, e))) \\ \mathrm{Bag}_E(e,X) \Leftrightarrow & v' \in X \leftrightarrow (\mathrm{Inc}(v', e) \vee (\exists e' \in E \setminus E_T)(\mathrm{Inc}(v', e') \\ & \wedge \mathrm{FundCyc}(e, e'))) \\ \mathrm{Parent}(X_p, X_c) \Leftrightarrow & \exists v(\exists e \in E_T)((\mathrm{Bag}_V(v, X_p) \wedge \mathrm{Bag}_E(e, X_c) \wedge \mathrm{head}(v, e)) \\ & \vee (\mathrm{Bag}_V(v, X_c) \wedge \mathrm{Bag}_E(e, X_p) \wedge \mathrm{tail}(v, e))) \\ \mathrm{nb}_<(X_a, X_b) \Leftrightarrow & (\exists e_a \in E_T)(\exists e_b \in E_T)(\mathrm{head}(e_a) = \mathrm{head}(e_b) \wedge \mathrm{nb}_<(e_a, e_b))\end{aligned}$$

A.9 k-Outerplanar Graphs

Using the forbidden minors (K_4 and $K_{2,3}$), we can define a predicate for verifying whether a graph is outerplanar in a straightforward way.

$$\text{Outerpl}(V', E') \Leftrightarrow \neg(\text{Minor}_{K_4}(V', E') \vee \text{Minor}_{K_{2,3}}(V', E'))$$

Following the argumentation in the proof of Lemma 6.7, we can define our predicate as follows.

$$\begin{aligned}\exists V_1 \cdots \exists V_k \Big(&\text{Part}_V(V, V_1, \ldots, V_k) \wedge \text{Outerpl}(V_1, \text{IncE}(V_1)) \\ &\wedge \cdots \wedge \text{Outerpl}(V_k, \text{IncE}(V_k)) \\ &\wedge \forall v \Big(\bigwedge_{i=1,\ldots,k} v \in V_i \rightarrow \forall w \forall e(\text{Edge}(e, v, w) \\ &\rightarrow (w \in V_{i-1} \vee w \in V_i \vee w \in V_{i+1}) \Big) \Big)\end{aligned}$$

A.9.1 3-Connected k-Outerplanar Graphs

We first give the necessary definition of defining the ordering $\text{nb}_<$ as described in Lemma 6.17. The first step is to define face-adjacency of two edges.

$$\begin{aligned}\text{Adj}_F(e, f) \Leftrightarrow &\exists v(\text{Inc}(v, e) \wedge \text{Inc}(v, f)) \\ &\wedge (\exists E' \subseteq E)(\text{FaceBd}_3(E') \wedge e \in E' \wedge f \in E')\end{aligned}$$

Next, we define a set to check whether a set of edges is a face-adjacency path from the one to the other, if they both share a vertex v. Intuitively speaking, this predicate states that each edge in the candidate set E' has precisely one neighbor in it, if the edge is either e or f and precisely two otherwise. Furthermore, E' has to consist of a subset of the incident edges of v, without $e'_{\mathcal{A}}$ (see the proof of Lemma 6.17) and it has to contain both e and f.

$$\begin{aligned}\text{Path}_F(E', e, f) \Leftrightarrow &(\exists E'' \subseteq (\text{IncE}(v) \setminus e'_{\mathcal{A}}))(E' = E'' \cup \{e, f\}) \\ &\wedge e_1 \in E' \leftrightarrow \Big(\Big((e_1 = e \vee e_1 = f) \wedge (\exists e_2 \in E')(\text{Adj}_F(e_1, e_2) \\ &\wedge (\forall e_3 \in E')((\neg e_2 = e_3) \rightarrow \neg \text{Adj}_F(e_1, e_3))) \Big) \\ &\vee \Big(\neg(e_1 = e \vee e_1 = f) \wedge (\exists e_2 \in E')(\exists e_3 \in E') \Big(\text{Adj}_F(e_1, e_2) \\ &\wedge \text{Adj}_F(e_1, e_3) \wedge (\forall e_4 \in E')((\neg(e_4 = e_2 \vee e_4 = e_3)) \\ &\rightarrow \neg \text{Adj}_F(e_1, e_4)) \Big) \Big) \Big)\end{aligned}$$

We are now ready to define the predicate for the ordering $\mathrm{nb}_<$.

$$\mathrm{nb}_<(e,f) \Leftrightarrow \exists E_e \exists E_f(\mathrm{Path}_F(E_e, e_{\mathcal{A}}, e) \wedge \mathrm{Path}_F(E_f, e_{\mathcal{A}}, f) \wedge E_e \subset E_f)$$

A.9.2 Tree Decompositions for 3-Connected k-Outerplanar Graphs

We first show how to define that a spanning tree with edge set F has bounded face remember number ν in a 3-connected planar graph $G = (V, E)$, which completes the proof of Proposition 6.20. Intuitively speaking, this predicate checks that for each combination of a vertex and a face boundary FB, the number edges, whose fundamental cycle uses both v and some edge in FB, is bounded by ν.

$$\begin{aligned} fr(V,E,F) \leq \nu \Leftrightarrow \forall v(\forall E_{FB} \subseteq E)\Big(&\mathrm{FaceBd}_3(E_{FB}) \rightarrow (\forall e_1 \in E \setminus F) \cdots (\forall e_{\nu+1} \in E \setminus F) \\ &\Big(\Big(\bigwedge_{1 \leq i \leq \nu+1} (\exists E_C \subseteq E)(\mathrm{FundCyc}(e_i, C_e) \wedge \neg(C_E \cap E_{FB} = \emptyset) \wedge \mathrm{Inc}(v, E_C))\Big) \\ &\rightarrow \bigvee_{1 \leq i < j \leq \nu+1} e_i = e_j\Big)\Big) \end{aligned}$$

Next, we will define the edge sets $C(v, f_i)$, as used in the proof of Lemma 6.19.

$$\begin{aligned} E' = C(v, E_{FB}, F) \Leftrightarrow e \in E' \leftrightarrow (\exists E_C \subseteq E)(&\mathrm{FundCyc}(e, E_C) \\ &\wedge \neg(E_C \cap E_{FB} = \emptyset) \wedge \mathrm{Inc}(v, E_C)) \end{aligned}$$

We furthermore denote by $C(v, e, F)$ the union of the sets $C(v, f_i)$ and $C(v, f_j)$ of the two faces f_i and f_j, whose face boundaries contain e (such that e is incident to v).

We now define a predicate identifying a unique face boundary with lowest layer number for each vertex.

$$\begin{aligned} \mathrm{Layer}_i(E_{FB}) \Leftrightarrow &\mathrm{FaceBd}_3(E_{FB}) \wedge \exists v(\mathrm{Inc}(v, E_{F_B}) \wedge v \in V_i) \\ E' = E_{f_\ell}(v) \Leftrightarrow &(\exists e \in E')\Big(\mathrm{Inc}(v,e) \wedge \bigwedge_{i=1,\ldots,k} \Big(v \in V_i \rightarrow \Big(\Big(\mathrm{Layer}_{i-1}(E') \\ &\wedge \neg((\exists f \exists E_f)(\mathrm{Layer}_{i-1}(E_f) \wedge f \in E_f \wedge \mathrm{Inc}(v,f) \wedge \mathrm{nb}_<(f,e))))\Big) \\ &\vee \Big(\mathrm{Layer}_i(E') \wedge \neg(\exists E_f(\mathrm{Layer}_{i-1}(E_f) \wedge \mathrm{Inc}(v, E_f))) \\ &\wedge \neg((\exists f \exists E_f)(\mathrm{Layer}_i(E_f) \wedge f \in E_f \wedge \mathrm{Inc}(v,f) \wedge \mathrm{nb}_<(f,e))))\Big)\Big)\Big) \end{aligned}$$

We are now ready to define the Bag-predicates of our tree decomposition. Note that the bag type σ can be defined in the same way as for bounded degree k-outerplanar graphs, hence we refer to Appendix A.8 for the details. The types σ_H can be defined using the predicates given above. We assume that we are given an arbitrary but fixed orientation

on the edges as described in the proof of Lemma 6.19.

$$\begin{aligned}\mathrm{Bag}_{\sigma_H}(e,X) \Leftrightarrow & v \in X \leftrightarrow \mathrm{head}(v,e) \vee (\exists e' \in (C(v,e,F) \cup C(v, E_{f_\ell}(\mathrm{head}(e)), F)) \\ & (\mathrm{Inc}(v,e') \wedge \forall w(\neg(v=w) \wedge \mathrm{Inc}(w,e')) \rightarrow \mathrm{col}(v) < \mathrm{col}(w))\end{aligned}$$

We can define the bag type σ_T by replacing 'head' by 'tail' in the above predicate.

We now define the set of anchor edges $E_{\mathcal{A}}$ and co-anchor edges $E'_{\mathcal{A}}$. For each vertex v we need to find a face with lowest layer number f_ℓ. Let e_{ℓ_1} and e_{ℓ_2} denote the incident edges of v bounding f_ℓ. Then, e_{ℓ_1} has to be contained in $E_{\mathcal{A}}$ and e_{ℓ_2} in $E'_{\mathcal{A}}$. Note that this choice is arbitrary and that we have to choose precisely one such face for each vertex in the graph.

$$\begin{aligned}E' = E_{\mathcal{A}} \Leftrightarrow & \forall v \exists e(e \in E' \wedge \mathrm{Inc}(v,e) \wedge e \in E_{f_\ell}(v) \\ & \wedge \forall e'((\mathrm{Inc}(v,e') \wedge \neg e = e') \rightarrow \neg(e' \in E'))) \\ E' = E'_{\mathcal{A}} \Leftrightarrow & (\forall e \in E_{\mathcal{A}}) \forall v \exists e'(e' \in E' \wedge \mathrm{Inc}(v,e) \wedge \mathrm{Inc}(v,e') \wedge e \in E_{f_\ell}(v) \wedge e' \in E_{f_\ell}(v) \\ & \wedge \forall e''((\mathrm{Inc}(v,e'') \wedge \neg e'' = e') \rightarrow \neg(e'' \in E')))\end{aligned}$$

We now turn to defining the Parent-predicate and begin by defining the case when a bag of type σ is a bag of type σ_T.

$$\begin{aligned}\mathrm{Parent}_{\sigma\sigma_T}(X,Y) \Leftrightarrow & (\exists e \in F)(\mathrm{Bag}_{\sigma}(e,X) \wedge \mathrm{Bag}_{\sigma_T}(e,Y)) \\ & \vee (\exists e \in F)(\exists e_\ell \in E_{f_\ell}(\mathrm{tail}(e)) \cap \mathrm{Inc}(\mathrm{tail}(e)))(\mathrm{Adj}_F(e,e_\ell) \\ & \wedge \mathrm{Bag}_{\sigma}(e_\ell,X) \wedge \mathrm{Bag}_{\sigma_T}(e,Y))\end{aligned}$$

Similarly, we can define the case when a bag of type σ_H is the parent of a bag of type σ.

$$\begin{aligned}\mathrm{Parent}_{\sigma_H\sigma}(X,Y) \Leftrightarrow & (\exists e \in F)(\mathrm{Bag}_{\sigma_H}(e,X) \wedge \mathrm{Bag}_{\sigma}(e,Y)) \\ & \vee (\exists e \in F)(\exists e_\ell \in E_{f_\ell}(\mathrm{head}(e)) \cap \mathrm{Inc}(\mathrm{head}(e)))(\mathrm{Adj}_F(e,e_\ell) \\ & \wedge \mathrm{Bag}_{\sigma_H}(e,X) \wedge \mathrm{Bag}_{\sigma}(e_\ell,Y))\end{aligned}$$

We now consider edges between bags of type σ_H/σ_T. In the following, we define the case when all bags involved are σ_T-bags and note that the other cases can be defined by the obvious replacements. We first define the outgoing edges of the σ_T-bag corresponding to the unique incoming edge in the directed spanning tree $T = (V,F)$.

$$\begin{aligned}\mathrm{Parent}^I_{\sigma_T\sigma_T}(X,Y) \Leftrightarrow & (\exists e^* \in F)(\exists e \in E)\Big(\mathrm{Bag}_{\sigma_T}(e^*,X) \wedge \mathrm{Bag}_{\sigma_T}(e,Y) \\ & \wedge \mathrm{tail}(e^*) = \mathrm{tail}(e) \wedge (\mathrm{nb}_{\prec}(e,e^*) \vee \mathrm{nb}_{\prec}(e^*,e))\Big)\end{aligned}$$

We now define the rest of the edges. We denote by $e^*(v)$ the edge which satisfies $e^* \in F \wedge \text{tail}(e^*) = v$.

$$\begin{aligned}\text{Parent}^R_{\sigma_T\sigma_T}(X,Y) \Leftrightarrow \exists e \exists f \Big(&\text{tail}(e) = \text{tail}(f) \wedge \text{nb}_{\prec}(e,f) \\ &\wedge \Big((\text{Bag}_{\sigma_T}(e,X) \wedge \text{Bag}_{\sigma_T}(f,Y) \wedge \text{nb}_{<}(e^*(\text{tail}(e)),e)) \\ &\vee (\text{Bag}_{\sigma_T}(f,X) \wedge \text{Bag}_{\sigma_T}(e,Y) \wedge \text{nb}_{<}(f,e^*(\text{tail}(f))))\Big)\Big)\end{aligned}$$

Unifying all above defined predicates (plus the omitted similar cases) yields the Parent(X,Y)-predicate for our tree decomposition.

A.10 Hierarchical Graph Decompositions for k-Outerplanar Graphs

In this section we provide details for the predicates used in proofs of Section 6.4. First we show how to define the parent-relation between blocks in our hierarchical decomposition as explained in the proof of Lemma 6.35. We assume that we are given a graph $G = (V,E)$ with a spanning tree $S = (V,F)$, which i rooted at an (arbitrary) vertex $r \in V$.

Let $Block(X)$ denote a predicate which is true if and only if a set $X \subseteq V$ is a block in the hierarchical decomposition of G. $Block(X)$ is definable by [14] (cf. also Proposition 6.36). Note that this predicate handles both the case when the parent is a cut-bag and the child is a block bag and vice versa (by the equivalence after the term '$(X \subset Y)$').

$$\begin{aligned}\text{Parent}_{Block}(X,Y) \Leftrightarrow &(Block(X) \wedge Block(Y) \wedge (X \cap Y = X \vee X \cap Y = Y)) \\ &\wedge (X \subset Y) \leftrightarrow \Big((\forall y \in Y)(\exists x \in X)(\forall P_x \subseteq F)(\forall P_y \subseteq F) \\ &\Big((\text{Path}_{\rightarrow}(r,x,P_x) \wedge \text{Path}_{\rightarrow}(r,y,P_y)) \rightarrow P_x \subset P_y \Big)\Big)\end{aligned}$$

A.10.1 Defining a Cycle Block

We now show how to define the predicates for tree decompositions of a cycle block $C = (W, E_C)$ as used in the proof of Proposition 6.37. First, we find the root $r_C \in W$ of the cycle.

$$\begin{aligned}v = r_C \Leftrightarrow &(r \in W \wedge v = r) \vee \Big((\exists C_P \subset V)(\text{Parent}_{Block}(C_P, C) \\ &\wedge \exists v((\text{Bag}_{\mathcal{C}_1}(v,C_P) \vee \text{Bag}_{\mathcal{C}_2}(v,C_P)) \wedge v = r_C))\Big)\end{aligned}$$

Now we can define the predicate Bag_{Cyc} straightforwardly.

$$\mathrm{Bag}_{Cyc}(e, X) \Leftrightarrow \neg\mathrm{Inc}(e, r_C) \wedge (v \in X \leftrightarrow (\mathrm{Inc}(v, e) \vee v = r_C))$$

Furthermore we can define the predicate $\mathrm{Parent}_{Cyc}(X, Y)$ as described in the proof of Proposition 6.37.

$$\begin{aligned}\mathrm{Parent}_{Cyc}(X, Y) \Leftrightarrow \exists e \exists f \Big(& \mathrm{Bag}_{Cyc}(e, X) \wedge \mathrm{Bag}_{Cyc}(f, Y) \wedge |X \cap Y| = 2 \\ & \wedge (\exists Z \subseteq V) \Big(\mathrm{Cycle}_{\mathcal{B}_3}(Z) \wedge \mathrm{Inc}(e, Z) \wedge \mathrm{Inc}(f, Z) \\ & \wedge (\exists P_e \subseteq \mathrm{IncE}(Z))(\exists P_f \subseteq \mathrm{IncE}(Z)) \\ & (\mathrm{Path}_{\to}(r_C, \mathrm{tail}(e), P_e) \wedge \mathrm{Path}_{\to}(r_C, f, P_f) \wedge P_e \subset P_f) \Big) \Big)\end{aligned}$$

A.10.2 Defining the Parent-predicate for $(\mathcal{T}, \mathcal{X})$

We now complete the proof of Lemma 6.35 by defining the parent-relation in all bags of the resulting tree decomposition $(\mathcal{T}, \mathcal{X})$ of the graph G. During this step we also modify some of the Bag-predicates, since, as explained in the proof, a number of vertices might be added to each bag in the tree decomposition. A vertex v is added to a bag X, when it is a member of a tree decomposition of a 2-connected 2-block or a 3-block and v is contained in the parent cut bag of X in the hierarchical decomposition of G. We show how to define such a predicate for an arbitrary case.

$$\begin{aligned}\mathrm{Bag}'_*(X) \Leftrightarrow (\exists X' \subseteq X) \Big(& \mathrm{Bag}_*(X') \wedge v \in X \setminus X' \leftrightarrow \exists Y \exists Z \Big(X' \subseteq Z \\ & \wedge (2\text{-}\mathrm{Conn}_{\mathcal{B}_2}(Z) \vee 3\text{-}\mathrm{Conn}_{\mathcal{B}_3}(Z) \vee \mathrm{Cycle}_{\mathcal{B}_3}(Z)) \\ & \wedge \mathrm{Parent}_{Block}(Y, Z) \wedge v \in Y \Big) \Big)\end{aligned}$$

In the following, we indicate that we refer to these modified bags by using the notation 'Bag′ . . .' instead of 'Bag . . .'. We define two cases: One, in which a $\mathcal{C}_1$- or $\mathcal{C}_2$-block is a parent of a $\mathcal{B}_3$-block and vice versa. The cases for $\mathcal{C}_1$- and $\mathcal{B}_2$-blocks can be defined by the obvious replacements. Note that the predicate $\mathrm{Root}_{\mathcal{B}_3}$ can be defined straightforwardly using the $\mathrm{Bag}_{\mathcal{B}_3}(X)$- and $\mathrm{Parent}_{\mathcal{B}_3}(X, Y)$-predicates.

$$\begin{aligned}\mathrm{Parent}_{\mathcal{C}\mathcal{B}_3}(X, Y) \Leftrightarrow & (\mathrm{Bag}'_{\mathcal{C}_1}(X) \vee \mathrm{Bag}'_{\mathcal{C}_2}(X)) \wedge \mathrm{Bag}'_{\mathcal{B}_3}(Y) \wedge X \subseteq Y \wedge \mathrm{Root}_{\mathcal{B}_3}(Y) \\ & \wedge \exists Z (Y \subseteq Z \wedge \mathrm{Parent}_{Block}(X, Z \setminus X)) \\ \mathrm{Parent}_{\mathcal{B}_3\mathcal{C}}(X, Y) \Leftrightarrow & \mathrm{Bag}'_{\mathcal{B}_3}(X) \wedge (\mathrm{Bag}'_{\mathcal{C}_2}(Y) \vee \mathrm{Bag}'_{\mathcal{C}_1}(Y)) \wedge X \subseteq Y \\ & \wedge \exists Z (X \subseteq Z \wedge \mathrm{Parent}_{Block}(X \setminus Z, Z)) \\ & \wedge \neg(\exists X' (\mathrm{Parent}_{\mathcal{B}_3}(X', X) \wedge X' \subseteq Y))\end{aligned}$$

The $\mathrm{Parent}_{\mathcal{BC}}(X, Y)$-predicate can now be defined as a unification of all these cases.

A.10.3 Defining Tree Decompositions for 3-Connected 3-Blocks

We now show how to define the predicates for defining the sets $\mathcal{S}_E$ and $\mathcal{R}_V$ as outlined in the proof of Lemma 6.44. To shorten our notation, we will use the symbol $E[B_3, \mathcal{S}_E]$ instead of the term 'IncE$(B_3) \cap \mathcal{S}_E$'.

(i) $(\exists \mathcal{R}_V \subseteq V)(\exists F \subseteq E)(\exists F' \subseteq E)(\exists \mathcal{S}_E \subseteq E)(\mathcal{S}_E = F \cup F') \ldots$

(ii) $(\exists r_T \in V)(\text{Tree}_{\rightarrow}(r_T, F)) \ldots$

(iii) $e \in F' \rightarrow \exists x \exists y (\neg x = y \wedge \text{Inc}(x, e) \wedge \text{Inc}(y, e) \wedge \neg e \in F$
$\wedge \exists X (\text{Bag}_{\mathcal{C}_2}(X) \wedge x \in X \wedge y \in X)) \ldots$

(iv) $(\forall B_3 \subseteq V)\Big(\text{3-Conn}_{\mathcal{B}_3}(B_3) \rightarrow \Big(er(B_3, \text{IncE}(B_3), E[B_3, \mathcal{S}_E]) \leq 2k$
$\wedge\, fr(B_3, \text{IncE}(B_3), E[B_3, \mathcal{S}_E]) \leq k\Big)\Big)$

(v) $v \in \mathcal{R}_V \rightarrow \exists X (\text{Bag}_{\mathcal{C}_2}(X) \wedge v \in X)$

(vi) $(\forall B_3 \subseteq V)\Big(\text{3-Conn}_{\mathcal{B}_3}(B_3) \rightarrow (\exists r_{B_3} \in \mathcal{R}_V)\Big(\text{Tree}_{\rightarrow}(r_{B_3}, B_3, E[B_3, \mathcal{S}_E])\Big)\Big)$