MSOL-Definability Equals Recognizability for Halin Graphs and Bounded Degree k-Outerplanar Graphs*

Lars Jaffke† and Hans L. Bodlaender‡

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 [8]. 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 prove this conjecture for a number of special cases in a stronger form. That is, we show that each recognizable property is definable in MSOL, i.e. the counting operation is not needed in our expressions. We give proofs for Halin graphs, bounded degree k-outerplanar graphs and some related graph classes. We furthermore show that the conjecture holds for any graph class that admits tree decompositions that can be defined in MSOL, thus providing a useful tool for future proofs.

1 Introduction

In a seminal paper from 1976, Rudolf Halin (1934-2014), lay the ground work for the notion of tree decompositions of graphs [13], which later was studied deeply in the proof of the famous Graph Minor Theorem by Robertson and Seymour [19] 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 [12].

Another seminal result is Courcelle's Theorem [8], 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

* The research of the second author was partially funded by the Networks programme, funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research.

† Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. Email: `l.jaffke@students.uu.nl`

‡ Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. Department of Mathematics and Computer Science, University of Technology Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. Email: `h.l.bodlaender@uu.nl`

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. [6]). 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 [8], which can be seen in the following example.

Example 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 [7]. 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 [16] is not considered to be valid by several experts. In the course of time proofs were given for the classes of trees and forests [8], partial 2-trees [9], partial 3-trees and k-connected partial k-trees [15]. A sketch of a proof for graphs of pathwidth at most k appeared at ICALP 1997 [14]. Very recently, one of the authors proved, in collaboration with Heggernes and Telle, that Courcelle's Conjecture holds for partial k-trees without chordless cycles of length at least ℓ [5].

In this paper we give self-contained proofs for Halin graphs, k-outerplanar graphs of bounded degree, a subclass of k-outerplanar graphs (of unbounded degree) and some classes related to feedback edge and/or vertex sets of bounded size w.r.t. a spanning tree in the graph. In all of these cases we show a somewhat stronger result, as we restrict ourselves to MSOL-definability, thus avoiding the above mentioned counting predicate. Since Halin graphs have treewidth 3 [21], Kaller's result implies that recognizable properties are CMSOL-definable in this case [15]. We strengthen this result to MSOL-definability.

Additionally, we show that Courcelle's Conjecture holds in our stronger sense for each graph class that admits certain types of MSOL-definable tree decompositions. We believe that this technique provides a useful tool towards its resolution — if not for all graph classes, then at least for a significant number of special cases.

In our proofs, we use another classic result in automata theory, the Myhill-Nerode Theory [17][18]. 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 [11, 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 [3].

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 the paper is organized as follows. In Section 2, we give the basic definitions and explain all concepts that we use in more detail. In Section 3 we prove some technical results regarding equivalence classes w.r.t. nodes in tree decompositions. The main results are presented in Sections 4 and 5, where we prove Courcelle's Conjecture for Halin graphs and other graph classes, such as bounded degree k-outerplanar graphs. We give some concluding remarks in Section 6.

2 Preliminaries

Graphs and Tree Decompositions

We begin by giving the basic definitions of the graph classes and some related concepts used throughout the paper.

Definition 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 (Halin Graph). *A graph is called a* Halin graph*, if it can be formed by a planar embedding of a tree, none of whose vertices has degree two, and a cycle that connects all leaves of the tree such that the embedding stays planar.*

Definition 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 exterior 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 1. *Halin graphs are 2-outerplanar graphs.*

The following definition will play a central role in many of the proofs of Sections 4 and 5.

Definition 4 (Fundamental Cycle). *Let $G = (V, E)$ be a graph with maximal spanning forest $T = (V, F)$. Given an edge $e = \{v, w\}$, $e \in E \backslash F$, 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 5 (Tree Decomposition, Treewidth). *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.*

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 6 (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.*
(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 7 (Rooted and Ordered Tree Decomposition). *Consider a tree decomposition $(T = (N, F), X)$. 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 8 (Terminal Graph). *A* terminal graph *$G = (V, E, X)$ is a graph with vertex set V, edge set E and an* ordered *terminal set $X \subseteq V$.*

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

Definition 9 ((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 $Y_{t'}$, $t, t' \in N$, 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 *$Y_{t'}$, denoted by $[X_t]^+_{Y_{t'}}$ 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 $Y_{t'}$, with terminal set X_t.*

The ordering in each terminal set of the above mentioned terminal graphs can be arbitrary, but fixed.

For an illustration of Definition 9, see Figure 1a, where $H = [X_H]^+$ and $G = [X_G]^+_{X_H}$.

Equivalence Relations

Definition 10 (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 the paper we will restrict ourselves to terminal graphs of fixed boundary size (i.e. the maximum size of terminal sets is bounded by some constant), 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]).

Definition 11 (Equivalence Relation over Terminal Graphs). *Let P denote a graph property. $\sim_P$ denotes an equivalence relation over terminal graphs,* describing *P, defined as follows. Let G, H and K be terminal graphs with fixed boundary size. Then we have:*

$$G \sim_P H \Leftrightarrow \forall K : P(G \oplus K) \Leftrightarrow P(H \oplus K)$$

This yields notions of equivalence classes *and* finite index *in the ordinary way. We might drop the index P in case it is clear from the context.*

We illustrate Definition 11 with an example.

Example 2. Let P denote the property that a graph has a Hamiltonian cycle. Let G and H be two terminal graphs with terminal sets X_G and X_H, respectively (where $|X_G| = |X_H|$). We say that G and H are equivalent w.r.t. $\sim_P$, if for all terminal graphs K (with terminal set X_K, $|X_K| = |X_G| = |X_H|$), 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.

As mentioned earlier, our ideas are based on the Myhill-Nerode Theory for graphs of bounded treewidth. The following theorem formally states this result.

Theorem 1 (Myhill-Nerode Theorem for Graphs of Treewidth k). *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) There exists an equivalence relation $\sim_P$, describing P, of finite index.

By the proof of this theorem (see, e.g., [11, Theorem 12.7.2]) we know that we can identify some equivalence classes of $\sim_P$ with accepting states in the automaton given in (i). Let C_P denote such an ('accepting') equivalence class and $G \in C_P$ a terminal graph. Then we know that the graph $G \oplus (X_G, \emptyset, X_G)$ has property P. We will use this fact in the proofs of Sections 4.3 and 5.1.

MSOL-Definability

We now define monadic second order logic over graphs. All variables that we use in our expressions are either single vertices/edges or vertex/edge sets. *Atomic predicates* are logical statements with the least number of variables, e.g. the vertex membership '$v \in V$'. Higher-order predicates can be formed by joining predicates via negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and equivalence $\leftrightarrow$ together with the existential quantifier $\exists$ and the universal quantifier $\forall$. A predicate without free variables, i.e. variables that are not in the scope of some quantifier, is called a *sentence*. A graph property is called *MSOL-definable* if we can express it with an MSOL-sentence.

A central concept used in this paper is an implicit representation of a tree decomposition in monadic second order logic, as we cannot refer to bags and edges in a tree decomposition as variables in MSOL directly. Hence, we most importantly require two types of predicates. 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 some *type*, whose definition depends on the actual graph class under consideration. 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.

While all MSOL-definable tree decompositions have to be rooted, not all of them have to be ordered. In some cases, however, an ordering on nodes with the same parent is another prerequisite, which also has to be verifiable with an MSOL-predicate.

Definition 12 (MSOL-definable tree decomposition). *A rooted (and ordered) tree decomposition (T, X) of a graph G is called* MSOL-definable*, if the following hold.*

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

(a) $Bag^V_{\tau_1}(v, X), \ldots, Bag^V_{\tau_t}(v, X)$*: The bag* X *is associated with type* τ_i *and the vertex* $v \in V$*, where* $1 \leq i \leq t$*.*
(b) $Bag^E_{\sigma_1}(e, X), \ldots, Bag^E_{\sigma_s}(e, X)$*: The bag* X *is associated with type* σ_j *and the edge* $e \in E$*, where* $1 \leq j \leq s$*.*

Furthermore there exists at least one type that contains the corresponding vertex or both endpoints of the corresponding edge.

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

We call an MSOL-definable tree decomposition ordered*, if the following holds.*

(iii) There exists a predicate $nb_{\prec}(X_l, X_r)$*, 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*.*

3 Constructing Equivalence Classes

The current section 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.

In the following, unless stated otherwise, we assume that our 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.

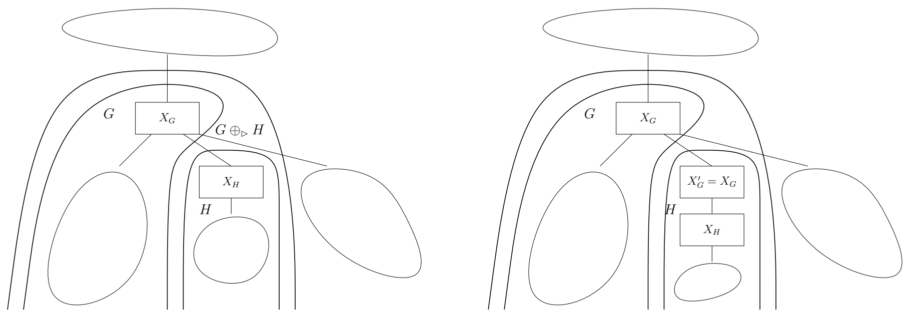
Definition 13 (Gluing via $\oplus_{\triangleright}$). *Let* X_G *be a branch bag in a tree decomposition with child* X_H *and let* $G = [X_G]^+_{X_H} = (V_G, E_G, X_G)$ *and* $H = [X_H]^+ = (V_H, E_H, X_H)$ *denote the partial terminal subgraph of* X_G *given* X_H *and the terminal subgraph of* X_H*, 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 1 and suppose that we know the equivalence class for the graph $G = [X_G]^+_{X_H}$, i.e. the partial terminal subgraph of X_G given X_H, and the equivalence class for graph $H = [X_H]^+$, the terminal subgraph of X_H. We want to derive the 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



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

Fig. 1. Branch node in a tree decomposition

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 (also up to isomorphism) completely unrelated to G^* and hence can be disregarded. The following lemma formalizes the above discussion.

Lemma 1. *Let X_G be a branch bag in a 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 14 (Gluing via $\oplus_T$). *Let G be a (terminal) graph and X an ordered set of vertices. The operation $\oplus_T$ is defined as:*

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

That is, we take the (not necessarily disjoint) union of X and the vertices in G and let X be the terminal set of the resulting terminal graph.

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 2. *Let G and H be two terminal graphs as in Lemma 1. Then,*

$$G \oplus_{\triangleright} H = \overbrace{(G \oplus \underbrace{(H \oplus_T X_G)}_{(b)}) \oplus_T X_G}^{(a)}. \tag{1}$$

This process of rewriting $\oplus_{\triangleright}$ can be illustrated as shown in Figure 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

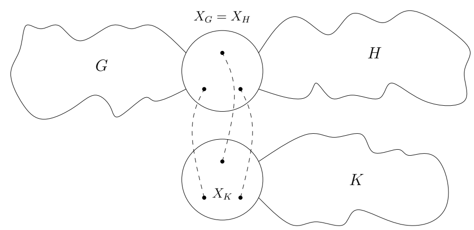


Fig. 2. Terminal graphs G, H and K as in the proof of Proposition 3. The dashed lines indicate, which vertices are being identified in the corresponding $\oplus$-operation.

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 1 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 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 1).

We now proceed with the formal proofs.

Proposition 3. *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 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 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. $\square$

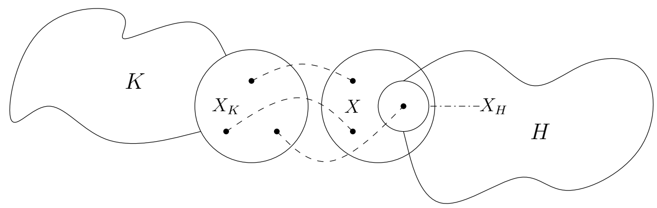


Fig. 3. Terminal graphs H and K, and a terminal set X. The dashed lines indicate, which vertices are being identified in the corresponding $\oplus$-operation.

Lemma 2. *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, one can derive a similar argument as in the proof of Proposition 3. 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.

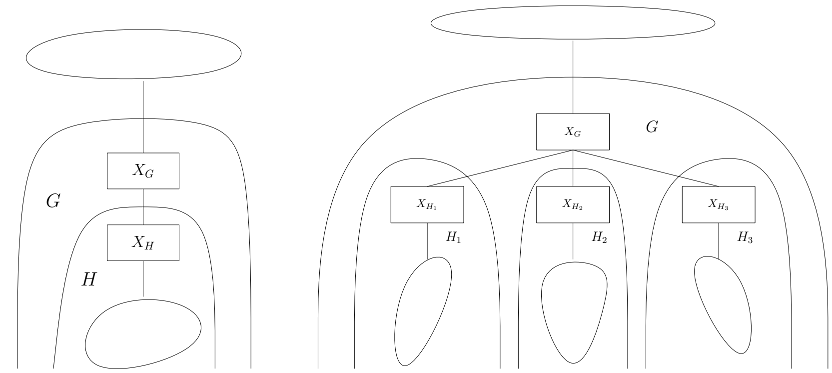
$$\begin{aligned}\forall K:\ & P(K \oplus (H \oplus_T X)) \Leftrightarrow P(H \oplus (K_X \oplus_T X_H)) \\ \Leftrightarrow & P(H' \oplus (K_X \oplus_T X_{H'})) \Leftrightarrow P(K \oplus (H' \oplus_T X))\end{aligned}$$

□

This conclude our proof of Lemma 1. □

The methods used in this proof also allow us to handle intermediate nodes in a tree decomposition. For an illustration see Figure 4a. Lemma 2 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 1, where we were dealing with two child nodes of a branch bag, to handle any constant number of children at a time (see Figure 4b). We will apply this result to tree decompositions that are not ordered but instead have bounded degree.



(a) Intermediate node, where $G = H \oplus_T X_G$.

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

Fig. 4. Intermediate and bounded degree branch node in a tree decomposition.

Lemma 3. *Let X_G be a branch bag in a tree decomposition with a constant number of child bags $X_{H_1}, \dots, X_{H_c}$. Let $H_1 = [X_{H_1}]^+, \dots,$ $H_c = [X_c]^+$. If $H_1' \sim H_1, \dots, H_c' \sim H_c$ and $X_{H_1'} = X_{H_1}, \dots, X_{H_c'} = X_{H_c}$, 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 2, 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 1, 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 $X_{H_1}, \dots, X_{H_c}$ 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). $\square$

4 Halin Graphs

This section 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 (or tree decomposition) properties, we use other graph properties that have been shown to be MSOL-definable before, 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 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 (Cf. [10], Lemma 4.8 in [15]). *Let G be a graph of treewidth k. Any orientation ϕ_{Ori} on its edges using predicates head(e, v) and tail(e, v) is MSOL-definable.*

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

(i) $\mathrm{col}(v) < \mathrm{col}(w)$ and $e \in F$, or
(ii) $\mathrm{col}(v) > \mathrm{col}(w)$ 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 head(e, v) and tail(e, v) as shown in Appendix A.1, we can define our sentence as

$$\exists X_0 \cdots \exists X_k (\exists F \subseteq E)(k+1\text{-col}(V, X_0, \ldots, X_k) \wedge \phi_{Ori}).$$

□

Lemma 5. *Let $G = (V, E)$ be a Halin graph. The orientation on the edge set of G such that its spanning tree forms a rooted directed tree and the outer cycle is a directed cycle, is MSOL-definable.*

Proof. Since Halin graphs have treewidth 3, we can use Lemma 4. 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

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

The MSOL-predicates given in Appendix A.1 complete the proof. □

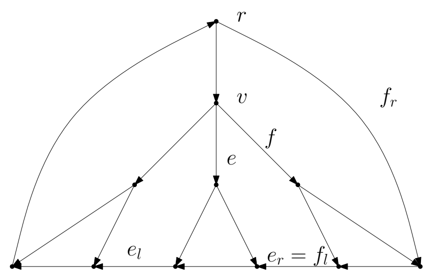


Fig. 5. Example of a Halin graph with edge orientation.

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 6 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 6. *For any vertex in a Halin graph there exists an ordering $nb_<$ on its child edges that is MSOL-definable.*

Proof. Let $G = (V, E)$ be a Halin graph with an orientation on its edges as shown in Lemma 5, E_T its edges of the spanning tree, 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 a Halin graph is contained in exactly two fundamental cycles. Assume 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 5 for an example.)

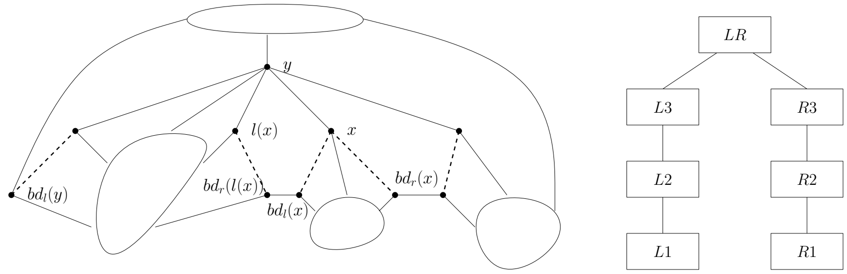
Now consider directed paths in E_C from r to the tail vertices of the above mentioned edges. If f is on the right-hand side of e, then the path from r to the tail of f_r is always the shortest of the four. The MSOL-predicates given in Appendix A.2 define such an ordering $\mathrm{nb}_<(e, f)$. □

4.2 MSOL-Definable Tree Decompositions

In this section we will describe how to construct a width-3 tree decomposition of a Halin graph that 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 15 (Left and Right Boundary Vertex). *Given a vertex $v \in V$ of a Halin graph G, a vertex is called its* left boundary vertex*, denoted by $bd_l(v)$*



(a) Structural overview of a Halin graph.

(b) The component created for each edge.

Fig. 6. Constructing a component of a tree decomposition for an edge of a Halin graph.

if 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 7. *Halin graphs admit width-3 MSOL-definable tree decompositions.*

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 6a 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 6b 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 indeed 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 $\mathrm{Parent}(X_p, X_c)$ 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.

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 1. *Halin graphs admit binary width-3 MSOL-definable tree decompositions such that all their leaf bags have size one.*

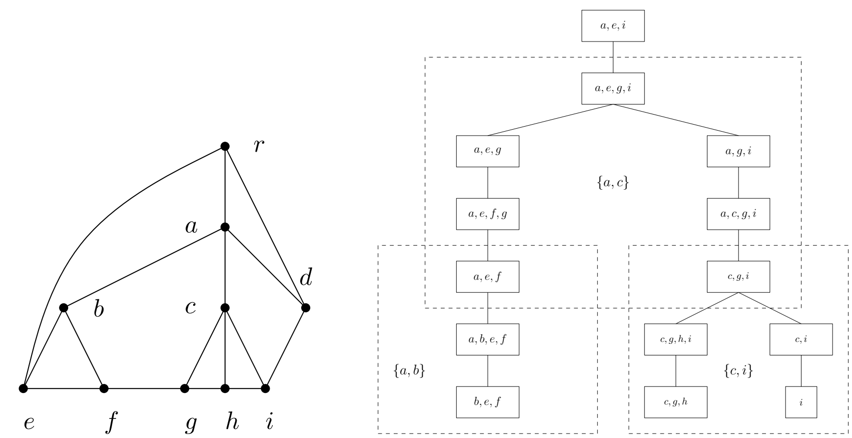
Proof. It is easy to see by the construction given in the proof of Lemma 7 that this tree decomposition is binary. All leaf bags are of type $R1$ and are associated with edges whose tail vertex x is a vertex on the outer cycle. Hence, $x = bd_l(x) = bd_r(x)$ and our claim follows. □

We will illustrate the construction of a tree decomposition given in the proof of Lemma 7 with the following example.

Example 3. Consider the graph depicted in Figure 7a. 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.
- $\{a, c\}$: 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 7b shows the resulting part of the tree decomposition.



(a) An example Halin graph. (b) The component of the tree decomposition corresponding to the denoted edges.

Fig. 7. An example subtree of a tree decomposition of a Halin graph.

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 8. *Finite index implies MSOL-definability for Halin graphs.*

Proof. By Lemma 7 we know that Halin graphs admit MSOL-definable tree decompositions of bounded width and thus 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 1 and 3 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 1).

Proposition 4. *There exist two functions $f_I : \mathbb{N} \times \mathcal{P}(V) \to \mathbb{N}$ and $f_J : \mathcal{P}_2(\mathbb{N}) \times \mathcal{P}(V) \to \mathbb{N}$, 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 1).

Next, we mimic the proof of Büchi's famous classic result for words over an alphabet [7], as shown in [20, 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 7) 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-sentence 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 1) 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, 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 in an equivalence class satisfying P, which we denote by ϕ_{Root}. We know that we can identify these equivalence classes by (the discussion given after) Theorem 1 and will denote them by $C_{A_1}, \ldots, C_{A_p}$.

Our MSOL-sentence then combines to:

$$\phi_{Leaf} \wedge \phi_{TSG} \wedge \phi_{Root} \tag{2}$$

Sentence 2 together with the details for the subsentences given in Appendix A.4 complete the proof. □

Combining Lemma 8 with Theorem 1 and [8], we directly obtain the following.

Theorem 2. *MSOL-definability equals recognizability for Halin graphs.*

5 Extensions

The methods we used in the proofs of Section 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 5.4. In the former we show that MSOL-definability equals recognizability for any graph class that admits either a bounded degree or an ordered MSOL-definable tree decomposition and in the latter we give the proof for bounded degree k-outerplanar graphs. Furthermore we study another subclass of k-outerplanar graphs in Section 5.2 and graphs that can be constructed with bounded size feedback edge and vertex sets in Section 5.3.

5.1 MSOL-Definable Tree Decompositions

We will now turn to generalizing the proof for Halin graphs to any graph class that admits MSOL-definable tree decompositions that are either ordered or have bounded degree. The proof works analogously as the proof of Lemma 8. This result will give us a useful tool to prove Courcelle's Conjecture for a number of graph classes, since it will follow immediately from the construction of MSOL-definable tree decompositions.

Lemma 9. *Finite index implies MSOL-definability for each graph class that admits MSOL-definable ordered tree decompositions of bounded width.*

Proof. It is easy to see that the predicate ϕ_{Root} can be defined in the same way as in the proof of Lemma 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 necessarily 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 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 assume that the equivalence classes of $\sim_P$ are denoted by $C_1, \ldots, C_r$. We can use the function f_I defined in Proposition 4 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 1.

Proposition 5. *There exists a function $f_J : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, 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:

1. Both X and Y correspond to a vertex.
2. Both X and Y correspond to an edge.
3. X corresponds to a vertex and Y to an edge.
4. X corresponds to an edge and Y to a vertex.

The predicates defining these cases for intermediate nodes are given in Appendix A.5.

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. Assume that a bag X is of type τ for 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. The predicates for branch nodes can be found in Appendix A.5, which complete the proof. $\square$

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 10. *Finite index implies MSOL-definability for each graph class that admits bounded degree MSOL-definable tree decompositions of bounded width.*

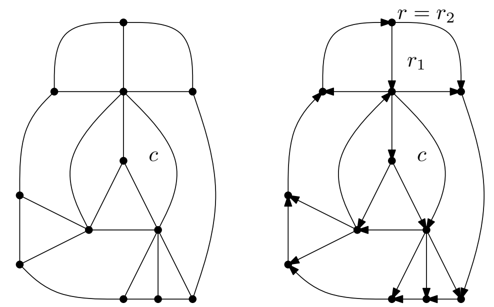
Proof. Since this proof works almost exactly as the proof of Lemma 9, we only state the differences. Let $c+1$ denote the maximum degree of a (branch) node in the tree decomposition and again we refer to the equivalence classes of $\sim_P$ as $C_1, \ldots, C_r$. Using Lemma 3 we know that the following holds (generalizing Proposition 4(ii)).

Proposition 6. *There exists a function $f_J : \mathcal{P}_c(\mathbb{N}) \times \mathcal{P}(V) \to \mathbb{N}$, 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 $f_J(\{c_1, \ldots, c_k\}, X)$.*

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 and our claim follows. $\square$

Combining Lemmas 9 and 10 with Theorem 1 and [8], we obtain the following.

Theorem 3. *MSOL-definability equals recognizability for graph classes that admit ordered or bounded degree MSOL-definable tree decompositions of width at most k.*



(a) G without edge orientation

(b) G with edge orientation

Fig. 8. An example 2-cycle tree G with central vertex c.

5.2 k-Cycle Trees

In this section we consider graph class which can be seen as a slight generalization of Halin graphs.

Definition 16 (k-cycle trees). *A graph G is called* cycle tree*, if it is a planar graph that 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. 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 8a shows an example of a 2-cycle tree. We easily observe the following.

Proposition 7. *Each k-cycle tree is k-outerplanar.*

Lemma 11. *Any edge orientation ϕ_{Ori} using predicates $head(e, v)$ and $tail(e, v)$ is MSOL-definable for k-outerplanar graphs .*

Proof. This follows immediately from Lemma 4 and the fact that k-outerplanar graphs have treewidth at most $3k - 1$ [4, Theorem 83]. $\square$

To prove our result for k-cycle trees, 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 17 (i-th boundary vertex). *Given a vertex v, we say that w is its* i-th left boundary vertex*, denoted by $bd^l_i(v)$, if w lies on C_i and there exists a path E^l_P from v to w, only using edges of the tree of the graph, 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^l_P. Similarly, we define the* i-th right boundary vertex *$bd^r_i(v)$.*

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

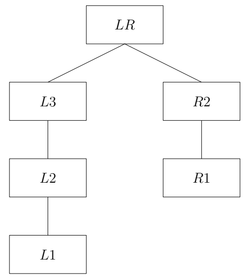


Fig. 9. Bag types and edges for a component in the tree decompositions of a k-cycle tree.

Lemma 12. *k-Cycle trees admit MSOL-definable binary tree decompositions of width at most $4k$.*

Proof. We can show this in almost exactly the same way as for Halin graphs (Lemma 7), 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 this edge orientation is MSOL-definable by Lemma 11 and the predicates given in Appendix A.6. For an illustration of the orientation see Figure 8b.

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 7, 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 $e = \{x, y\}$ such that y is the parent of x and y lies on cycle C_i, we have the following types of bags, with edges between them as shown in Figure 9. (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 ver-

tices 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 $4k$. The predicates in Appendix A.6 complete the proof. □

Combining Lemma 12 with Theorem 3, we can derive the following.

Theorem 4. *MSOL-definability equals recognizability for k-cycle trees.*

5.3 Feedback Edge and Vertex Sets

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

Definition 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. *Let $G = (V, E)$ be a graph with spanning tree $T = (V, F)$, which admits an MSOL-definable (ordered) tree decomposition of width k, 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 MSOL-definable (ordered) tree decomposition of width $k + l$, 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. ◻

As an example we apply Theorem 5 to both Halin graphs and k-cycle trees, which - in combination with Theorem 3 - yields the following result.

Theorem 6. *Let $\mathcal{C}$ denote a graph class such that its members can be constructed from a Halin graph or a k-cycle tree together with either an edge set or vertex set as described in Theorem 5. Then, MSOL-definability equals recognizability for all members of $\mathcal{C}$.*

5.4 Bounded Degree k-Outerplanar Graphs

We now give another method for proving Courcelle's conjecture based on the notion of vertex and edge remember numbers, which will enable us to prove it for k-outerplanar graphs of bounded degree. We first give the necessary definitions.

Definition 19 (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 that use v. Analogously, we define the* edge remember number*, denoted by $er(G, T)$.*

Theorem 7. *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) a width-k MSOL-definable tree decomposition of bounded degree, if G has bounded degree.
(ii) a width-k MSOL-definable ordered tree decomposition, if there is an MSOL-definable ordering $nb_<(e, f)$ over all edges $e, f \in F$ with the same head vertex.

Proof. For both (i) and (ii) 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(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,[1] so we can directly define such predicates, see Appendix A.8.

For case (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.

Case (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 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 13 (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 $vr(G, T) \leq 3k - 1$.*

Given the nature of its proof, one immediately has the following consequence.

Corollary 2. *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 8. *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 2, 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 7(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 3. □

Note that the theorem also holds, if we add feedback edge and vertex sets to a k-outerplanar graph of bounded degree, as explained in Theorem 5.

[1] This clearly holds by Lemma 4, since trees have treewidth 1.

6 Conclusion

In this paper we showed that MSOL-definability equals recognizability for Halin graphs, k-cycle trees, graph classes constructed using certain feedback edge or vertex sets and bounded degree k-outerplanar graphs. Hence we proved a number of special cases of Courcelle's Conjecture [8], which states that each graph property that is recognizable for graphs of bounded treewidth is CMSOL-definable, additionally strengthening it to MSOL-definability.

For our proofs, we introduced the concept of MSOL-definable tree decompositions, and used MSOL-definable tree decompositions of bounded degree or ordered MSOL-definable tree decompositions (i.e. admitting an ordering on nodes with the same parent). We additionally showed that this conjecture holds for any graph class that admits either one of these kinds of tree decompositions.

We hope that the techniques of our paper give useful tools to solve other special cases in the future, and also help to establish the border between cases that allow MSOL-definability versus cases that need the counting predicate of CMSOL.

We plan to further investigate the case of k-outerplanar graphs and believe that the following conjecture holds.

Conjecture 1. Recognizability equals

(i) MSOL-definability for 3-connected k-outerplanar graphs.
(ii) CMSOL-definability for k-outerplanar graphs.

We also hope to establish that 3-connectedness is a necessary condition to avoid the counting predicate in our proof, which for k-outerplanar graphs will provide us a with clear separation between MSOL and CMSOL.

Another interesting graph property that might be used in such proofs is Hamiltonicity (in our sense that means a graph admits a Hamiltonian *path*). It is easy to see that one can order nodes with the same parent in an MSOL-definable tree decomposition, if the underlying graph admits a Hamiltonian path, hence we conjecture the following.

Conjecture 2. MSOL-definability equals recognizability for (3-connected) Hamiltonian partial k-trees.

Acknowledgements

The second author thanks Bruno Courcelle, Mike Fellows, Pinar Heggernes and Jan Arne Telle for inspiring discussions.

References

[1] Abrahamson, K.R., Fellows, M.R.: 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. Contemporary Mathematics, vol. 147, pp. 539–564. AMS (1993)

[2] Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. Journal of Algorithms 12(2), 308–340 (1991)
[3] van Bevern, R., Fellows, M.R., Gaspers, S., Rosamond, F.A.: Myhill-Nerode methods for hypergraphs. In: Proceedings ISAAC 2013, LNCS, vol. 8283, pp. 372–382. Springer (2013)
[4] Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science 209(1-2), 1–45 (1998)
[5] Bodlaender, H.L., Heggernes, P., Telle, J.A.: paper in preparation (2015)
[6] Borie, R.B., Parker, R.G., Tovey, C.A.: Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. Algorithmica 7(1-6), 555–581 (1992)
[7] Büchi, J.R.: Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly 6(1-6), 66–92 (1960)
[8] Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Information and Computation 85(1), 12–75 (1990)
[9] Courcelle, B.: The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. Theoretical Computer Science 80(2), 153–202 (1991)
[10] Courcelle, B.: The monadic second-order logic of graphs VIII: Orientations. Annals of Pure and Applied Logic 72(2), 103–143 (1995)
[11] Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer (2013)
[12] Halin, R.: Studies on minimally n-connected graphs. Combinatorial Mathematics and its applications pp. 129–136 (1971)
[13] Halin, R.: S-functions for graphs. Journal of Geometry 8(1-2), 171–186 (1976)
[14] Kabanets, V.: Recognizability equals definability for partial k-paths. In: Proceedings ICALP 1997, LNCS, vol. 1256, pp. 805–815. Springer (1997)
[15] Kaller, D.: Definability equals recognizability of partial 3-trees and k-connected partial k-trees. Algorithmica 27(3-4), 348–381 (2000)
[16] Lapoire, D.: Recognizability equals monadic second-order definability for sets of graphs of bounded tree-width. In: Proceedings STACS 1998, LNCS, vol. 1373, pp. 618–628. Springer (1998)
[17] Myhill, J.R.: Finite automata and the representation of events. Tech. Rep. WADC TR-57-624, Wright-Paterson Air Force Base (1957)
[18] Nerode, A.: Linear automaton transformations. Proceedings of the American Mathematical Society 9(4), 541–544 (1958)
[19] Robertson, N., Seymour, P.: Graph minors. III. Planar tree-width. Journal of Combinatorial Theory, Series B 36(1), 49–64 (1984)
[20] Thomas, W.: Languages, automata, and logic. In: Handbook of Formal Languages. Beyond Words. vol. 3, pp. 389–455. Springer (1996)
[21] Wimer, T.V.: Linear Algorithms on K-terminal Graphs. Ph.D. 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. [6], 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 $v, w, v', w', \dots$ typically represent vertices and $e, f, e', f', \dots$ 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 [6]). 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 [6]), 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)
- $\mathrm{Conn}(V, E)$, $\mathrm{Conn}_k(V, E)$, $\mathrm{Cycle}(V, E)$, $\mathrm{Tree}(V, E)$, $\mathrm{Path}(V, E)$

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 Lemma 5. 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.

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

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 [6, 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-$\mathrm{reg}_\leftarrow$, k-$\mathrm{reg}_\rightarrow$ and k-$\mathrm{reg}_\leftrightarrow$, respectively). This enables us to define predicates for directed trees and cycles.

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

A.2 Child Ordering of a Halin Graph

This section concludes the proof of Lemma 6, that is we define an ordering on edges in a Halin graph that have the same parent in the tree E_T. Therefore we define predicates for directed paths and fundamental cycles. Note that $\mathrm{Path}_{\to}(s,t,E')$ is true if and only if E' is a directed $s-t$-path.

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

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 6.

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

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

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

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 7, 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.

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.

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

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

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

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

Bag Types We define an MSOL-predicate for each bag type that we introduced in the proof of Lemma 7. Using the definition of boundary vertices given above, we can define them in a straightforward manner.

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

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.

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

The Parent Relation We now turn to defining the predicate $\mathrm{Parent}(X_p, X_c)$, 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 6b and the respective parent/child relationships between components.

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

A.4 Equivalence Class Membership for Halin Graphs

In this section we complete the proof of Lemma 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 $R1$.

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

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.

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

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 $\text{nb}_{\prec}$-ordering or one of the edges is the parent edge of the other one.

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

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.

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

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

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

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 9. 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 10.

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.

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

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 9 w.r.t. the types of the bags X and Y.

$$\begin{aligned}\phi_{TSG,Int} =&\forall X\forall Y\Big((\mathrm{Int}(X)\wedge\mathrm{Parent}(X,Y))\\ &\to\bigwedge_{\substack{\tau,\tau'\in\{\tau_1,\ldots,\tau_t\}\\ \sigma,\sigma'\in\{\sigma_1,\ldots,\sigma_s\}}}\Big(\phi_{Int,\tau,\tau'}\wedge\phi_{Int,\sigma,\sigma'}\wedge\phi_{Int,\tau,\sigma}\wedge\phi_{Int,\sigma,\tau}\Big)\Big)\end{aligned}\tag{3}$$

Case 1. 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.

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

Case 2. 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.

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

Case 3. 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:

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

Case 4. The bag X belongs to an edge and Y belongs to a vertex. For σ, τ as above we define:

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

Branch Nodes In the following we will define predicates for branch nodes, 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.

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

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 $[X]^+_{Y'}$.

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

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.

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

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.

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

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 3), for all combinations of vertex/edge types.

Branch Nodes for Bounded Degree Tree Decompositions To finish the proof of Lemma 10, 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 $X_1, \ldots, X_k$, all corresponding to vertices in the graph and types $\tau_1, \ldots, \tau_k$. Then we can define this predicate as follows.

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

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 12. 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.

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

We can write down the orientation ϕ_{Ori} described in the proof of Lemma 12 in the following way.

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

We can define a predicate $\mathrm{nb}^i_<(e,f)$ in complete analogy to $\mathrm{nb}_<(e,f)$ 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 17).

$$\begin{aligned} w = bd^r_i(v, E') \Leftrightarrow & (w \in V_{C_i} \wedge v = w) \\ & \vee (\exists E_P \subseteq E')(\mathrm{Path}_{\rightarrow}(v, w, E_P) \wedge w \in \mathrm{IncV}(E_{C_i})) \\ & \wedge (\forall e \in E_P) \neg (\exists E'_P \subseteq E') \Big(\mathrm{Path}_{\rightarrow}(v, w, E'_P) \wedge (\exists e' \in E'_P) \\ & \bigvee_{i=1,\ldots,k} \mathrm{nb}^i_<(e, e') \Big) \end{aligned} \tag{4}$$

To define bd^l_i, we simply replace $\mathrm{nb}^i_<(e,e')$ by $\mathrm{nb}^i_<(e',e)$ in line 4. 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.

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

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

$$\begin{aligned} \mathrm{Bag}_{R1}(e, X) \Leftrightarrow & z \in X \leftrightarrow \bigvee_{i<j\leq k} \Big(z = bd^l_j(x) \vee z = bd^r_j(x) \Big) \\ \mathrm{Bag}_{R2}(e, X) \Leftrightarrow & z \in X \leftrightarrow \Big(z = y \vee \bigvee_{i<j\leq k} \Big(z = bd^l_j(x) \vee z = bd^r_j(x) \Big) \Big) \end{aligned}$$

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

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

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. In the following, let $G' = (V', E')$ and $G = (V, E)$ be graphs as stated in Theorem 5. 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).[2]

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

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

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

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 7, 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.

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

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

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

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.

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