Extension Complexity, MSO Logic, and Treewidth*

Petr Kolman[1], Martin Koutecký[2], and Hans Raj Tiwary[3]

1 **Department of Applied Mathematics (KAM) & Institute of Theoretical Computer Science (ITI), Faculty of Mathematics and Physics (MFF), Charles University, Prague, Czech Republic**
kolman@kam.mff.cuni.cz

2 **Department of Applied Mathematics (KAM) & Institute of Theoretical Computer Science (ITI), Faculty of Mathematics and Physics (MFF), Charles University, Prague, Czech Republic**
koutecky@kam.mff.cuni.cz

3 **Department of Applied Mathematics (KAM) & Institute of Theoretical Computer Science (ITI), Faculty of Mathematics and Physics (MFF), Charles University, Prague, Czech Republic**
hansraj@kam.mff.cuni.cz

Abstract

We consider the convex hull $P_\varphi(G)$ of all satisfying assignments of a given MSO_2 formula φ on a given graph G. We show that there exists an extended formulation of the polytope $P_\varphi(G)$ that can be described by $f(|\varphi|, \tau) \cdot n$ inequalities, where n is the number of vertices in G, τ is the treewidth of G and f is a computable function depending only on φ and τ.

In other words, we prove that the extension complexity of $P_\varphi(G)$ is linear in the size of the graph G, with a constant depending on the treewidth of G and the formula φ. This provides a very general yet very simple meta-theorem about the extension complexity of polytopes related to a wide class of problems and graphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.4.1 Mathematical Logic, G.1.6 Optimization, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Extension Complexity, FPT, Courcelle's Theorem, MSO Logic

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.18

1 Introduction

In the '70s and '80s, it was repeatedly observed that various NP-hard problems are solvable in polynomial time on graphs resembling trees. The graph property of *resembling a tree* was eventually formalized as having bounded treewidth, and in the beginning of the '90s, the class of problems efficiently solvable on graphs of bounded treewidth was shown to contain the class of problems definable by the Monadic Second Order Logic (MSO_2) (Courcelle [11], Arnborg et al. [1], Courcelle and Mosbah [13]). Using similar techniques, analogous results for weaker logics were then proven for wider graph classes such as graphs of bounded cliquewidth and rankwidth [12]. Results of this kind are usually referred to as *Courcelle's theorem* for a specific class of structures.

In this paper we study the class of problems definable by the MSO logic from the perspective of extension complexity. While small extended formulations are known for various special classes of polytopes, we are not aware of any other result in the theory of

* This research was partially supported by projects GA15-11559S of GA ČR and 338216 of GA UK.

extended formulations that works on a wide class of polytopes the way Courcelle's theorem works for a wide class of problems and graphs.

Our Contribution. We prove that satisfying assignments of an MSO_2 formula φ on a graph of bounded treewidth can be *expressed* by a "small" linear program. More precisely, there exists a computable function f such that the convex hull – $P_\varphi(G)$ – of satisfying assignments of φ on a graph G on n vertices with treewidth τ can be obtained as the projection of a polytope described by $f(|\varphi|, \tau) \cdot n$ linear inequalities; we call $P_\varphi(G)$ the *MSO polytope*. All our results can be extended to general finite structures where the restriction on treewidth applies to the treewidth of their Gaifman graph [30].

Our proof essentially works by "merging the common wisdom" from the areas of extended formulations and fixed parameter tractability. It is known that dynamic programming can be turned into a compact extended formulation [32, 18], and that Courcelle's theorem can be seen as an instance of dynamic programming [26]; therefore one can expect the polytope of satisfying assignments of an MSO formula of a bounded treewidth graph to be compact.

However, there are a few roadblocks in trying to merge these two folklore wisdoms. For one, while Courcelle's theorem being an instance of dynamic programming in some sense may be obvious to an FPT theorist, it is far from clear to anyone else what that sentence may even mean. On the other hand, being able to turn a dynamic program into a compact polytope may be a theoretical possibility for an expert on extended formulations, but it is by no means an easy statement for an outsider to comprehend. What complicates the matters even further is that the result of Martin et al. [32] is not a result that can be used in a black box fashion. That is, a certain condition must be satisfied to get a compact extended formulation out of a dynamic program. This is far from a trivial task, especially for a theorem like Courcelle's theorem.

The rest of the article is organized as follows. In Section 2 we review some previous work related to Courcelle's theorem and extended formulations. In Section 3 we describe the relevant notions related to polytopes, extended formulations, graphs, treewidth and MSO logic. In Section 4 we prove the existence of compact extended formulations for MSO polytopes parameterized by the length of the given MSO formula and the treewidth of the given graph. In Section 5 we describe how to efficiently construct such a polytope given a tree decomposition of a graph.

2 Related Work

2.1 MSO Logic vs. Treewidth

Because of the wide relevance of the treewidth parameter in many areas (cf. the survey of Bodlaender [5]) and the large expressivity of the MSO and its extensions (cf. the survey of Langer et al. [27]), considerable attention was given to Courcelle's theorem by theorists from various fields, reinterpreting it into their own setting. These reinterpretations helped uncover several interesting connections.

The classical way of proving Courcelle's theorem is constructing a tree automaton A in time only dependent on φ and the treewidth τ, such that A accepts a tree decomposition of a graph of treewidth τ if and only if the corresponding graph satisfies φ; this is the automata theory perspective [11]. Another perspective comes from finite model theory where one can prove that a certain equivalence on the set of graphs of treewidth at most τ has only finitely many (depending on φ and τ) equivalence classes and that it *behaves well* [16]. Another approach proves that a quite different equivalence on so-called extended model checking games has finitely many equivalence classes [23] as well; this is the game-theoretic perspective. It can be observed that the finiteness in either perspective stems from the same roots.

Another related result is an *expressivity* result: Gottlob et al. [16] prove that on bounded treewidth graphs, a certain subset of the database query language Datalog has the same expressive power as the MSO. This provides an interesting connection between the automata theory and the database theory.

2.2 Extended Formulations

Sellmann, Mercier, and Leventhal [34] claimed to show compact extended formulation for binary Constraint Satisfaction Problems (CSP) for graphs of bounded treewidth, but their proof is not correct [33]. The first two authors of this paper gave extended formulations for CSP that has polynomial size for instances whose constraint graph has bounded treewidth [25] using a different technique. Bienstock and Munoz [3] prove similar results for the approximate and exact version of the problem. In the exact case, Bienstock and Munoz's bounds are slightly worse than those of Kolman and Koutecký [25]. It is worth noting that CSPs are a restricted subclass of problems that can be modeled using MSO logic. Laurent [28] provides extended formulations for the independent set and max cut polytopes of size $O(2^\tau n)$ for n-vertex graphs of treewidth τ and, independently, Buchanan and Butenko [8] provide an extended formulation for the independent set polytope of the same size.

A lot of recent work on extended formulations has focussed on establishing lower bounds in various settings: exact, approximate, linear vs. semidefinite, etc. (See for example [15, 2, 6, 29]). A wide variety of tools have been developed and used for these results including connections to nonnegative matrix factorizations [37], communication complexity [14], information theory [7], and quantum communication [15] among others.

For proving upper bounds on extended formulations, several authors have proposed various tools as well. Kaibel and Loos [19] describe a setting of branched polyhedral systems which was later used by Kaibel and Pashkovich [20] to provide a way to construct polytopes using reflection relations.

A particularly specific composition rule, which we term glued product (cf. Subsection 3.1), was studied by Margot in his PhD thesis [31]. Margot showed that a property called the projected face property suffices to glue two polytopes efficiently. Conforti and Pashkovich [10] describe and strengthen Margot's result to make the projected face property to be a necessary and sufficient condition to describe the glued product in a particularly efficient way.

Martin et al. [32] have shown that under certain conditions, an efficient dynamic programming based algorithm can be turned into a compact extended formulation. Kaibel [18] summarizes this and various other methods.

3 Preliminaries

3.1 Polytopes, Extended Formulations and Extension Complexity

For background on polytopes we refer the reader to Grünbaum [17] and Ziegler [38]. To simplify reading of the paper for the audience that is not working often in the area of polyhedral combinatorics, we provide here a brief glossary of common polyhedral notions that are used in this article.

A *hyperplane* in $\mathbb{R}^n$ is a closed convex set of the form $\{x | a^\intercal x = b\}$ where $a \in \mathbb{R}^n, b \in \mathbb{R}$. A *halfspace* in $\mathbb{R}^n$ is a closed convex set of the form $\{x | a^\intercal x \leqslant b\}$ where $a \in \mathbb{R}^n, b \in \mathbb{R}$. The inequality $a^\intercal x \leqslant b$ is said to define the corresponding halfspace. A *polytope* $P \subseteq \mathbb{R}^n$

is a bounded subset defined by intersection of finite number of halfspaces. A result of Minkowsky-Weyl states that equivalently, every polytope is the convex hull of a finite number of points. Let h be a halfspace defined by an inequality $a^\intercal x \leqslant b$; the inequality is said to be *valid* for a polytope P if $P = P \cap h$. Let h be a halfspace defined by a valid inequality $a^\intercal x \leqslant b$; then, $P \cap \{x | a^\intercal x = b\}$ is said to be a *face* of P.

Note that, taking a to be the zero vector and $b = 0$ results in the face being P itself. Also, taking a to be the zero vector and $b = 1$ results in the empty set. These two faces are often called the *trivial faces* and they are polytopes "living in" dimensions n and -1, respectively. Every face - that is not trivial - is itself a polytope of dimension d where $0 \leqslant d \leqslant n-1$. The zero dimensional faces of a polytope are called its *vertices*, and the $(n-1)$-dimensional faces are called its *facets*.

It is not uncommon to refer to three separate (but related) objects as a face: the actual face as defined above, the valid inequality defining it, and the equation corresponding to the valid inequality. While this is clearly a misuse of notation, the context usually makes it clear as to exactly which object is being referred to.

Let P be a polytope in $\mathbb{R}^d$. A polytope Q in $\mathbb{R}^{d+r}$ is called an *extended formulation* or an *extension* of P if P is a projection of Q onto the first d coordinates. Note that for any linear map $\pi : \mathbb{R}^{d+r} \to \mathbb{R}^d$ such that $P = \pi(Q)$, a polytope Q' exists such that P is obtained by dropping all but the first d coordinates on Q' and, moreover, Q and Q' have the same number of facets.

The *size* of a polytope is defined to be the number of its facet-defining inequalities. Finally, the *extension complexity* of a polytope P, denoted by $\mathrm{xc}(P)$, is the size of its smallest extended formulation. We refer the readers to the surveys [9, 35, 18, 36] for details and background of the subject and we only state three basic propositions about extended formulations here.

▶ **Proposition 1.** *Let P be a polytope with a vertex set $V = \{v_1, \ldots, v_n\}$. Then $\mathrm{xc}(P) \leqslant n$.*

Proof. Let $P = \mathrm{conv}(\{v_1, \ldots, v_n\})$ be a polytope. Then, P is the projection of

$$Q = \left\{ (x, \lambda) \,\middle|\, x = \sum_{i=1}^{n} \lambda_i v_i; \sum_{i=1}^{n} \lambda_i = 1; \lambda_i \geqslant 0 \text{ for } i \in \{1, \ldots, n\} \right\}.$$

It is clear that Q has at most n facets and therefore $\mathrm{xc}(P) \leqslant n$. ◀

▶ **Proposition 2.** *Let P be a polytope obtained by intersecting a set H of hyperplanes with a polytope Q. Then $\mathrm{xc}(P) \leqslant \mathrm{xc}(Q)$.*

Proof. Note that any extended formulation of Q, when intersected with H, gives an extended formulation of P. Intersecting a polytope with hyperplanes does not increase the number of facet-defining inequalities (and only possibly reduces it). ◀

The (cartesian) *product* of two polytopes P_1 and P_2 is defined as

$$P_1 \times P_2 = \mathrm{conv}\left(\{(x, y) \mid x \in P_1, y \in P_2\}\right).$$

▶ **Proposition 3.** *Let P_1, P_2 be two polytopes. Then*

$$\mathrm{xc}(P_1 \times P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2) \;.$$

Proof. Let Q_1 and Q_2 be extended formulations of P_1 and P_2, respectively. Then, $Q_1 \times Q_2$ is an extended formulation of $P_1 \times P_2$. Now assume that $Q_1 = \{x \mid Ax \leqslant b\}$ and $Q_2 = \{y \mid Cy \leqslant d\}$ and that these are the smallest extended formulations of P_1 and P_2, resp. Then

$$Q_1 \times Q_2 = \{(x, y) \mid Ax \leqslant b, Cy \leqslant d\} \ .$$

That is, we have an extended formulation of $P_1 \times P_2$ of size at most $\mathrm{xc}(P_1) + \mathrm{xc}(P_2)$. ◀

We are going to define the glued product of polytopes, a slight generalization of the usual product of polytopes. We use a case where the extension complexity of the glued product of two polytopes is upper bounded by the sum of the extension complexities of the two polytopes, and use it in Section 4 to describe a small extended formulation for the MSO polytope $P_\varphi(G)$ on graphs with bounded treewidth.

Let $P \subseteq \mathbb{R}^{d_1+k}$ and $Q \subseteq \mathbb{R}^{d_2+k}$ be 0/1-polytopes defined by m_1 and m_2 inequalities and with vertex sets $\mathrm{vert}(P)$ and $\mathrm{vert}(Q)$, respectively. Let $I_P \subseteq \{1, \ldots d_1 + k\}$ be a subset of coordinates of size k, $I_Q \subseteq \{1, \ldots d_2 + k\}$ be a subset of coordinates of size k, and let $I'_P = \{1, \ldots d_1 + k\} \setminus I_P$. For a vector x, and a subset I of coordinates, we denote by $x|_I$ the subvector of x specified by the coordinates I. The *glued product* of P and Q, (glued) with respect to the k coordinates I_P and I_Q, denoted by $P \times_k Q$, is defined as

$$P \times_k Q = \mathrm{conv}\left(\left\{(x|_{I'_P}, y) \in \mathbb{R}^{d_1+d_2+k} \mid x \in \mathrm{vert}(P), y \in \mathrm{vert}(Q), x|_{I_P} = y|_{I_Q}\right\}\right) .$$

We adopt the following convention while discussing glued products in the rest of this article. In the above scenario, we say that $P \times_k Q$ is obtained by gluing P and Q along the k coordinates I_P of P with the k coordinates I_Q of Q. If, for example, these coordinates are named z in P and w in Q, then we also say that P and Q have been glued along the z and w coordinates and we refer to the coordinates z and w as the *glued coordinates.* In the special case that we glue along the last k coordinates, the definition of the glued product simplifies to

$$P \times_k Q = \mathrm{conv}\left(\{(x, y, z) \in \mathbb{R}^{d_1+d_2+k} \mid (x, z) \in \mathrm{vert}(P), (y, z) \in \mathrm{vert}(Q)\}\right) .$$

This notion was studied by Margot [31] who provided a sufficient condition for being able to write the glued product in a specific (and efficient) way from the descriptions of P and Q. We will use this particular way in Lemma 1. The existing work [31, 10], however, is more focused on characterizing exactly when this particular method works. We do not need the result in its full generality and therefore we only state a very specific version of it that is relevant for our purposes; for the sake of completeness, we also provide a proof of it.

▶ **Lemma 1** (Gluing lemma). *Let P and Q be 0/1-polytopes and let the k (glued) coordinates in P be labeled $z_1, \ldots, z_k$, and the k (glued) coordinates in Q be labeled $w_1, \ldots, w_k$. Suppose that $\mathbf{1}^\intercal z \leqslant 1$ is valid for P and $\mathbf{1}^\intercal w \leqslant 1$ is valid for Q. Then $\mathrm{xc}(P \times_k Q) \leqslant \mathrm{xc}(P) + \mathrm{xc}(Q)$.*

Proof. Let (x', z', y', w') be a point from $P \times Q \cap \{(x, z, y, w) | z = w\}$. Observe that the point (x', z') is a convex combination of points $(x', 0), (x', e_1), \ldots, (x', e_k)$ from P with coefficients $(1 - \sum_{i=1}^k z'_i), z'_1, z'_2, \ldots, z'_k$ where e_i is the i-th unit vector. Similarly, the point (y', w') is a convex combination of points $(y', 0), (y', e_1), \ldots, (y', e_k)$ from Q with coefficients $(1 - \sum_{i=1}^k w'_i), w'_1, w'_2, \ldots, w'_k$. Notice that for every $j \in [k]$, (x'_j, e_j, y'_j) is a point from the glued product. As $w_i = z_i$ for every $i \in [k]$, we conclude that $(x', w', z') \in P \times_k Q$. Thus, by Proposition 2 the extension complexity of $P \times_k Q$ is at most that of $P \times Q$ which is at most $\mathrm{xc}(P) + \mathrm{xc}(Q)$ by Proposition 3. ◀

3.2 Graphs and Treewidth

For notions related to the treewidth of a graph and nice tree decomposition, in most cases we stick to the standard terminology as given in the book by Kloks [22]; the only deviation is in the leaf nodes of the nice tree decomposition where we assume that the bags are empty. For a vertex $v \in V$ of a graph $G = (V, E)$, we denote by $\delta(v)$ the set of neighbors of v in G, that is, $\delta(v) = \{u \in V \mid \{u, v\} \in E\}$.

A *tree decomposition* of a graph $G = (V, E)$ is a tree T in which each *node* $a \in T$ has an assigned set of vertices $B(a) \subseteq V$ (called a *bag*) such that $\bigcup_{a \in T} B(a) = V$ with the following properties:

- for any $\{u, v\} \in E$, there exists a node $a \in T$ such that $u, v \in B(a)$.
- if $v \in B(a)$ and $v \in B(b)$, then $v \in B(c)$ for all nodes c on the path from a to b in T.

The *treewidth $tw(T)$ of a tree decomposition* T is the size of the largest bag of T minus one. The *treewidth $tw(G)$ of a graph* G is the minimum treewidth over all possible tree decompositions of G.

A *nice tree decomposition* is a tree decomposition with one special node r called the *root* in which each node is one of the following types:

- *Leaf node*: a leaf a of T with $B(a) = \emptyset$.
- *Introduce node*: an internal node a of T with one child b for which $B(a) = B(b) \cup \{v\}$ for some $v \in B(a)$.
- *Forget node*: an internal node a of T with one child b for which $B(a) = B(b) \setminus \{v\}$ for some $v \in B(b)$.
- *Join node*: an internal node a with two children b and c with $B(a) = B(b) = B(c)$.

For a vertex $v \in V$, we denote by *top*(v) the topmost node of the nice tree decomposition T that contains v in its bag. For any graph G on n vertices, a nice tree decomposition of G with at most $8n$ nodes can be computed in time $\mathcal{O}(n)$ [4, 22].

Given a graph $G = (V, E)$ and a subset of vertices $\{v_1, \ldots, v_d\} \subseteq V$, we denote by $G[v_1, \ldots, v_d]$ the subgraph of G induced by the vertices $v_1, \ldots, v_d$. Given a tree decomposition T and a node $a \in V(T)$, we denote by T_a the subtree of T rooted in a, and by G_a the subgraph of G induced by all vertices in bags of T_a, that is, $G_a = G[\bigcup_{b \in V(T_a)} B(b)]$. Throughout this paper we assume that for every graph, its vertex set is a subset of $\mathbb{N}$. We define the following operator σ: for any set $U = \{v_1, v_2, \ldots, v_l\} \subseteq \mathbb{N}$, $\sigma(U) = (v_{i_1}, v_{i_2}, \ldots, v_{i_l})$ such that $v_{i_1} < v_{i_2} \cdots < v_{i_l}$.

For an integer $m \geq 0$, an $[m]$*-colored graph* is a pair $(G, \vec{V})$ where $G = (V, E)$ is a graph and $\vec{V} = (V_1, \ldots, V_m)$ is an m-tuple of subsets of vertices of G called an $[m]$*-coloring of* G. For integers $m \geq 0$ and $\tau \geq 0$, an $[m]$*-colored τ-boundaried graph* is a triple $(G, \vec{V}, \vec{p})$ where $(G, \vec{V})$ is an $[m]$-colored graph and $\vec{p} = (p_1, \ldots, p_\tau)$ is a τ-tuple of vertices of G called a *boundary* of G. If the tuples $\vec{V}$ and $\vec{p}$ are clear from the context or if their content is not important, we simply denote an $[m]$-colored τ-boundaried graph by $G^{[m],\tau}$. For a tuple $\vec{p} = (p_1, \ldots, p_\tau)$, we denote by p the corresponding set, that is, $p = \{p_1, \ldots, p_\tau\}$.

Two $[m]$-colored τ-boundaried *graphs* $(G_1, \vec{V}, \vec{p})$ and $(G_2, \vec{U}, \vec{q})$ are *compatible* if the function $h : \vec{p} \to \vec{q}$, defined by $h(p_i) = q_i$ for each i, is an isomorphism of the induced subgraphs $G_1[p_1, \ldots, p_\tau]$ and $G_2[q_1, \ldots, q_\tau]$, and if for each i and j, $p_i \in V_j \Leftrightarrow q_i \in U_j$.

Given two compatible $[m]$-colored τ-boundaried graphs $G_1^{[m],\tau} = (G_1, \vec{U}, \vec{p})$ and $G_2^{[m],\tau} = (G_2, \vec{W}, \vec{q})$, the *join* of $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$, denoted by $G_1^{[m],\tau} \oplus G_2^{[m],\tau}$, is the $[m]$-colored τ-boundaried graph $G^{[m],\tau} = (G, \vec{V}, \vec{p})$ where

- G is the graph obtained by taking the disjoint union of G_1 and G_2, and for each i, identifying the vertex p_i with the vertex q_i and keeping the label p_i for it;
- $\vec{V} = (V_1, \ldots, V_m)$ with $V_j = U_j \cup W_j$ and every q_i replaced by p_i, for each j;

- $\vec{p} = (p_1, \ldots, p_\tau)$ with p_i being the node in $V(G)$ obtained by the identification of $p_i \in V(G_1)$ and $q_i \in V(G_2)$, for each i.

Because of the choice of referring to the boundary vertices by their names in $G_1^{[m],\tau}$, it does not always hold that $G_1^{[m],\tau} \oplus G_2^{[m],\tau} = G_2^{[m],\tau} \oplus G_1^{[m],\tau}$; however, the two structures are isomorphic and equivalent for our purposes (see below).

3.3 Monadic Second Order Logic and Types of Graphs

In most cases, we stick to standard notation as given by Libkin [30]. A *vocabulary* σ is a finite collection of *constant* symbols $c_1, c_2, \ldots$ and *relation* symbols $P_1, P_2, \ldots$. Each relation symbol P_i has an associated arity r_i. A *σ-structure* is a tuple $\mathcal{A} = (A, \{c_i^{\mathcal{A}}\}, \{P_i^{\mathcal{A}}\})$ that consists of a universe A together with an interpretation of the constant and relation symbols: each constant symbol c_i from σ is associated with an element $c_i^{\mathcal{A}} \in A$ and each relation symbol P_i from σ is associated with an r_i-ary relation $P_i^{\mathcal{A}} \subseteq A^{r_i}$.

To give an example, a graph $G = (V, E)$ can be viewed as a σ_1-structure $(V, \emptyset, \{E\})$ where E is a symmetric binary relation on $V \times V$ and the vocabulary σ_1 contains a single relation symbol. Alternatively, for another vocabulary σ_2 containing three relation symbols, one of arity two and two of arity one, one can view a graph $G = (V, E)$ also as a σ_2-structure $I(G) = (V_I, \emptyset, \{E_I, L_V, L_E\})$, with $V_I = V \cup E$, $E_I = \{\{v, e\} \mid v \in e, e \in E\}$, $L_V = V$ and $L_E = E$; we will call $I(G)$ the *incidence graph* of G. In our approach we will make use of the well known fact that the treewidths of G and $I(G)$, viewed as a σ_1- and σ_2- structures as explained above, differ by one at most [24].

The main subject of this paper are formulas for graphs in monadic second order logic (MSO) which is an extension of first order logic that allows quantification over monadic predicates (i.e., over sets of vertices). By MSO$_2$ we denote the extension of MSO that allows in addition quantification over sets of edges. As every MSO$_2$ formula φ over σ_1 can be turned into an MSO formula φ' over σ_2 such that for every graph G, $G \models \varphi$ if and only if $I(G) \models \varphi'$ [folklore], for the sake of presentation we restrict our attention, without loss of generality, to MSO formulae over the σ_2 vocabulary. To further simplify the presentation, without loss of generality (cf. [21]) we assume that the input formulae are given in a variant of MSO that uses only set variables (and no element variables).

An important kind of structures that are necessary in the proofs in this paper are the $[m]$-colored τ-boundaried graphs. An $[m]$-colored τ-boundaried graph $G = (V, E)$ with boundary $p_1, \ldots, p_\tau$ colored with $V_1, \ldots, V_m$ is viewed as a structure $(V_I, \{p_1, \ldots, p_\tau\}, \{E_I, L_V, L_E, V_1, \ldots, V_m\})$; for notational simplicity, we stick to the notation $G^{[m],\tau}$ or $(G, \vec{V}, \vec{p})$. The corresponding vocabulary is denoted by $\sigma_{m,\tau}$.

A variable X is *free* in φ if it does not appear in any quantification in φ. If $\vec{X}$ is the tuple of all free variables in φ, we write $\varphi(\vec{X})$. A variable X is *bound* in φ if it is not free. By $qr(\varphi)$ we denote the *quantifier rank* of φ which is the number of quantifiers of φ when transformed into the prenex form (i.e., all quantifiers are at the beginning of the formula). We denote by MSO$[k, \tau, m]$ the set of all MSO formulae φ over the vocabulary $\sigma_{\tau,m}$ with $qr(\varphi) \leq k$.

Two $[m]$-colored τ-boundaried graphs $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$ are MSO$[k]$*-elementarily equivalent* if they satisfy the same MSO$[k, \tau, m]$ formulae; this is denoted by $G_1^{[m],\tau} \equiv_k^{MSO} G_2^{[m],\tau}$. The main tool in the model theoretic approach to Courcelle's theorem, that will also play a crucial role in our approach, can be stated as the following theorem.

▶ **Theorem 2** (follows from Proposition 7.5 and Theorem 7.7 [30]). *For any fixed $\tau, k, m \in \mathbb{N}$, the equivalence relation $\equiv_k^{MSO}$ has a finite number of equivalence classes.*

Let us denote the equivalence classes of the relation $\equiv_k^{MSO}$ by $\mathcal{C} = \{\alpha_1 \ldots, \alpha_w\}$, fixing an ordering such that α_1 is the class containing the empty graph. Note that the size of $\mathcal{C}$ depends only on k, m and τ, that is, $|\mathcal{C}| = f(k, m, \tau)$ for some computable function f. For a given MSO formula φ with m free variables, we define an *indicator function* $\rho_\varphi : \{1, \ldots, |\mathcal{C}|\} \to \{0, 1\}$ as follows: for every i, if there exists a graph $G^{[m],\tau} \in \alpha_i$ such that $G^{[m],\tau} \models \varphi$, we set $\rho_\varphi(i) = 1$, and we set $\rho_\varphi(i) = 0$ otherwise; note that if there exists a graph $G^{[m],\tau} \in \alpha_i$ such that $G^{[m],\tau} \models \varphi$, then $G'^{[m],\tau} \models \varphi$ for every $G'^{[m],\tau} \in \alpha_i$.

For every $[m]$-colored τ-boundaried graph $G^{[m],\tau}$, its *type*, with respect to the relation $\equiv_k^{MSO}$, is the class to which $G^{[m],\tau}$ belongs. We say that *types* α_i and α_j are *compatible* if there exist two $[m]$-colored τ-boundaried graphs of types α_i and α_j that are compatible; note that this is well defined as all $[m]$-colored τ-boundaried graphs of a given type are compatible. For every $i \geq 1$, we will encode the type α_i naturally as a binary vector $\{0, 1\}^{|\mathcal{C}|}$ with exactly one 1, namely with 1 on the position i.

An important property of the types and the join operation is that the type of a join of two $[m]$-colored τ-boundaried graphs depends on their types only.

▶ **Lemma 3** (Lemma 7.11 [30] and Lemma 3.5 [16]). *Let $G_a^{[m],\tau}$, $G_{a'}^{[m],\tau}$, $G_b^{[m],\tau}$ and $G_{b'}^{[m],\tau}$ be $[m]$-colored τ-boundaried graphs such that $G_a^{[m],\tau} \equiv_k^{MSO} G_{a'}^{[m],\tau}$ and $G_b^{[m],\tau} \equiv_k^{MSO} G_{b'}^{[m],\tau}$. Then $(G_a^{[m],\tau} \oplus G_b^{[m],\tau}) \equiv_k^{MSO} (G_{a'}^{[m],\tau} \oplus G_{b'}^{[m],\tau})$.*

The importance of the lemma rests in the fact that for determination of the type of a join of two $[m]$-colored τ-boundaried graphs, it suffices to know only a *small* amount of information about the two graphs, namely their types. The following two lemmas deal in a similar way with the type of a graph in other situations.

▶ **Lemma 4** (implicitly in [16]). *Let $(G_a, \vec{X}, \vec{p})$, $(G_b, \vec{Y}, \vec{q})$ be $[m]$-colored τ-boundaried graphs and let $(G_{a'}, \vec{X'}, \vec{p'})$, $(G_{b'}, \vec{Y'}, \vec{q'})$ be $[m]$-colored $(\tau+1)$-boundaried graphs with $G_a = (V, E)$, $G_{a'} = (V', E')$, $G_b = (W, F)$, $G_{b'} = (W', F')$ such that for some $v \notin V$ and $w \notin W$*

1. *$(G_a, \vec{X}, \vec{p}) \equiv_k^{MSO} (G_b, \vec{Y}, \vec{q})$;*
2. *$V' = V \cup \{v\}$, $\delta(v) \subseteq p$, $\vec{p}$ is a subtuple of $\vec{p'}$ and $(G_{a'}[V], \vec{X'}[V], \vec{p'}[V]) = (G_a, \vec{X}, \vec{p})$;*
3. *$W' = W \cup \{w\}$, $\delta(w) \subseteq q$, $\vec{q}$ is a subtuple of $\vec{q'}$ and $(G_{b'}[W], \vec{Y'}[W], \vec{q'}[W]) = (G_b, \vec{Y}, \vec{q})$;*
4. *$(G_{a'}, \vec{X'}, \vec{p'})$ and $(G_{b'}, \vec{Y'}, \vec{q'})$ are compatible.*

Then $(G_{a'}, \vec{X'}, \vec{p'}) \equiv_k^{MSO} (G_{b'}, \vec{Y'}, \vec{q'})$.

▶ **Lemma 5** (implicitly in [16]). *Let $(G_a, \vec{X}, \vec{p})$, $(G_b, \vec{Y}, \vec{q})$ be $[m]$-colored τ-boundaried graphs and let $(G_{a'}, \vec{X'}, \vec{p'})$, $(G_{b'}, \vec{Y'}, \vec{q'})$ be $[m]$-colored $(\tau+1)$-boundaried graphs with $G_a = (V, E)$, $G_{a'} = (V', E')$, $G_b = (W, F)$, $G_{b'} = (W', F')$ such that*

1. *$(G_{a'}, \vec{X'}, \vec{p'}) \equiv_k^{MSO} (G_{b'}, \vec{Y'}, \vec{q'})$;*
2. *$V \subseteq V'$, $|V'| = |V| + 1$, $\vec{p}$ is a subtuple of $\vec{p'}$ and $(G_{a'}[V], \vec{X'}[V], \vec{p'}[V]) = (G_a, \vec{X}, \vec{p})$;*
3. *$W \subseteq W'$, $|W'| = |W| + 1$, $\vec{q}$ is a subtuple of $\vec{q'}$ and $(G_{b'}[W], \vec{Y'}[W], \vec{q'}[W]) = (G_b, \vec{Y}, \vec{q})$.*

Then $(G_a, \vec{X}, \vec{p}) \equiv_k^{MSO} (G_b, \vec{Y}, \vec{q})$.

3.4 Feasible Types

Suppose that we are given an MSO formula φ over σ_2 with m free variables and a quantifier rank at most k, a graph G of treewidth at most τ, and a nice tree decomposition T of the graph G.

For every node of T we are going to define certain types and tuples of types as *feasible*. For a node $b \in V(T)$ of any kind (leaf, introduce, forget, join) and for $\alpha \in \mathcal{C}$, we say that α is a *feasible type of the node b* if there exist an $[m]$-coloring $\vec{X} = (X_1, \ldots, X_m)$ of G_b such that $(G_b, \vec{X}, \sigma(B(b)))$ is of type α; we say that $\vec{X}$ *realizes type* α *on the node* b. We denote the set of feasible types of the node b by $\mathcal{F}(b)$.

For an *introduce* node $b \in V(T)$ with a child $a \in V(T)$ (assuming that v is the new vertex), for $\alpha \in \mathcal{F}(a)$ and $\beta \in \mathcal{F}(b)$, we say that (α, β) is a *feasible pair of types for* b if there exist $\vec{X} = (X_1, \ldots, X_m)$ and $\vec{X}' = (X'_1, \ldots, X'_m)$ realizing types α and β on the nodes a and b, respectively, such that for each i, either $X'_i = X_i$ or $X'_i = X_i \cup \{v\}$. We denote the set of feasible pairs of types of the introduce node b by $\mathcal{F}_p(b)$.

For a *forget* node $b \in V(T)$ with a child $a \in V(T)$ and for $\beta \in \mathcal{F}(b)$ and $\alpha \in \mathcal{F}(a)$, we say (α, β) is a *feasible pair of types for* b if there exists $\vec{X}$ realizing β on b and α on a. We denote the set of feasible pairs of types of the forget node b by $\mathcal{F}_p(b)$.

For a *join* node $c \in V(T)$ with children $a, b \in V(T)$ and for $\alpha \in \mathcal{F}(c)$, $\gamma_1 \in \mathcal{F}(a)$ and $\gamma_2 \in \mathcal{F}(b)$, we say that $(\gamma_1, \gamma_2, \alpha)$ is a *feasible triple of types for* c if γ_1, γ_2 and α are mutually compatible and there exist $\vec{X}^1, \vec{X}^2$ realizing γ_1 and γ_2 on a and b, respectively, such that $\vec{X} = (X^1_1 \cup X^2_1, \ldots, X^1_m \cup X^2_m)$ realizes α on c. We denote the set of feasible triples of types of the join node c by $\mathcal{F}_t(c)$.

We define an *indicator function* $\mu : \mathcal{C} \times V(G) \times \{1, \ldots, m\} \to \{0, 1\}$ such that $\mu(\beta, v, i) = 1$ if and only if there exists $\vec{X} = (X_1, \ldots, X_m)$ realizing the type β on the node $top(v) \in V(T)$ and $v \in X_i$.

4 Extension Complexity of the MSO Polytope

For a given MSO formula $\varphi(\vec{X})$ over σ_2 with m free set variables $X_1, \ldots, X_m$, we define a polytope of satisfying assignments on a given graph G, represented as a σ_2 structure $I(G) = (V_I, \emptyset, \{E_I, L_V, L_E\})$ with domain of size n, in a natural way. We encode any assignment of elements of $I(G)$ to the sets $X_1, \ldots, X_m$ as follows. For each X_i in φ and each v in V_I, we introduce a binary variable y^v_i. We set y^v_i to be one if $v \in X_i$ and zero otherwise. For a given $0/1$ vector y, we say that y *satisfies* φ if interpreting the coordinates of y as described above yields a satisfying assignment for φ. The polytope of satisfying assignments, also called the *MSO polytope*, is defined as

$$P_\varphi(G) = \operatorname{conv}\left(\{y \in \{0,1\}^{nm} \mid y \text{ satisfies } \varphi\}\right).$$

▶ **Theorem 6** (Extension Complexity of the MSO Polytope). *For every graph G and for every MSO formula φ, $\operatorname{xc}(P_\varphi(G)) = f(|\varphi|, \tau) \cdot n$ where f is some computable function, $\tau = tw(G)$ and $n = |V_I|$.*

Proof. Let T be a fixed nice tree decomposition of treewidth τ of the given graph G represented as $I(G)$ and let k denote the quantifier rank of φ and m the number of free variables of φ. Let $\mathcal{C}$ be the set of equivalence classes of the relation $\equiv^{MSO}_k$. For each node b of T we introduce $|\mathcal{C}|$ binary variables that will represent a feasible type of the node b; we denote the vector of them by t_b (i.e., $t_b \in \{0,1\}^{|\mathcal{C}|}$). For each introduce and each forget node b of T, we introduce additional $|\mathcal{C}|$ binary variables that will represent a feasible type of the child (descendant) of b; we denote the vector of them by d_b (i.e., $d_b \in \{0,1\}^{|\mathcal{C}|}$). Similarly, for each join node b we introduce additional $|\mathcal{C}|$ binary variables, denoted by l_b, that will represent a feasible type of the left child of b, and other $|\mathcal{C}|$ binary variables, denoted by r_b, that will represent a feasible type of the right child of b (i.e., $l_b, r_b \in \{0,1\}^{|\mathcal{C}|}$).

We are going to describe inductively a polytope in the dimension given (roughly) by all the binary variables of all nodes of the given nice tree decomposition. Then we show that its extension complexity is small and that a properly chosen face of it is an extension of $P_\varphi(G)$.

First, for each node b of T, depending on its type, we define a polytope P_b as follows:

- b is a *leaf*. P_b consists of a single point $P_b = \{\overbrace{100\ldots0}^{|\mathcal{C}|}\}$.
- b is an *introduce* or *forget* node. For each feasible pair of types $(\alpha_i, \alpha_j) \in \mathcal{F}_p(b)$ of the node b, we create a vector $(d_b, t_b) \in \{0,1\}^{2|\mathcal{C}|}$ with $d_b[i] = t_b[j] = 1$ and all other coordinates zero. P_b is defined as the convex hull of all such vectors.
- b is a *join* node. For each feasible triple of types $(\alpha_h, \alpha_i, \alpha_j) \in \mathcal{F}_t(b)$ of the node b, we create a vector $(l_b, r_b, t_b) \in \{0,1\}^{3|\mathcal{C}|}$ with $l_b[h] = r_b[i] = t_b[j] = 1$ and all other coordinates zero. P_b is defined as the convex hull of all such vectors.

It is clear that for every node b in T, the polytope P_b contains at most $|\mathcal{C}|^3$ vertices, and, thus, by Proposition 1 it has extension complexity at most $\mathrm{xc}(P_b) \leqslant |\mathcal{C}|^3$. Recalling our discussion in Section 3 about the size of $\mathcal{C}$, we conclude that there exists a function f such that for every $b \in V(T)$, it holds that $\mathrm{xc}(P_b) \leqslant f(|\varphi|, \tau)$.

To obtain an extended formulation for $P_\varphi(G)$, we first glue these polytopes together, starting in the leaves of T and processing T in a bottom up fashion. We create polytopes Q_b for each node b in T recursively as follows:

- If b is a leaf then $Q_b = P_b$.
- If b is an introduce or forget node, then $Q_b = Q_a \times_{|\mathcal{C}|} P_b$ where a is the child of b and the gluing is done along the coordinates t_a in Q_a and d_b in P_b.
- If b is a join node, then we first define $R_b = Q_a \times_{|\mathcal{C}|} P_b$ where a is the left child of b and the gluing is done along the coordinates t_a in Q_a and l_b in P_b. Then Q_b is obtained by gluing R_b with Q_c along the coordinates t_c in Q_c and r_b in R_b where c is the right child of b.

The following lemma states the key property of the polytopes Q_b's.

▶ **Lemma 7.** *For every vertex y of the polytope Q_b there exist an $[m]$-coloring $\vec{X} = (X_1, \ldots, X_m)$ of G_b such that $(G_b, \vec{X}, \sigma(B(b)))$ is of type α where α is the unique type such that the coordinate of y corresponding to the binary variable $t_b(\alpha)$ is equal to one.*

Proof. The proof is by induction, starting in the leaves of T and going up towards the root. For leaves, the lemma easily follows from the definition of the polytopes P_b's.

For the inductive step, we consider an inner node b of T and we distinguish three cases:

- If b is a join node, then the claim for b follows from the inductive assumptions for the children of b, definition of a feasible triple, definition of the polytope P_b, Lemma 3 and the construction of the polytope Q_b.
- If b is an introduce node or a forget node, respectively, then, analogously, the claim for b follows from the inductive assumption for the child of b, definition of a feasible pair, definition of the polytope P_b, Lemma 4 or Lemma 5, respectively, and the construction of the polytope Q_b. ◀

Let c be the root node of the tree decomposition T. Consider the polytope Q_c. From the construction of Q_c, our previous discussion and the Gluing lemma, it follows that $\mathrm{xc}(Q_c) \leqslant \sum_{b \in V(T)} \mathrm{xc}(P_b) \leqslant f(|\varphi|, \tau) \cdot \mathcal{O}(n)$. It remains to show that a properly chosen face of Q_c is an extension of $P_\varphi(G)$. We start by observing that $\sum_{i=1}^{|\mathcal{C}|} t_c[i] \leq 1$ and $\sum_{i=1}^{|\mathcal{C}|} \rho_\varphi(i) \cdot t_c[i] \leq 1$, where ρ_φ is the indicator function, are valid inequalities for Q_c.

Let Q_φ be the face of Q_c corresponding to the valid inequality $\sum_{i=1}^{|\mathcal{C}|} \rho_\varphi(i) \cdot t_c[i] \leq 1$. Then, by Lemma 7, the polytope Q_φ represents those $[m]$-colorings of G for which φ holds. The corresponding feasible assignments of φ on G are obtained as follows: for every vertex $v \in V(G)$ and every $i \in \{1, \ldots, m\}$ we set $y_i^v = \sum_{j=1}^{|\mathcal{C}|} \mu(\alpha_j, v, i) \cdot t_{top(v)}[j]$. The sum is 1 if and only if there exists a type j such that $t_{top(v)}[j] = 1$ and at the same time $\mu(\alpha_j, v, i) = 1$; by the definition of the indicator function μ in Subsection 3.4, this implies that $v \in X_i$. Thus, by applying the above projection to Q_φ we obtain $P_\varphi(G)$, as desired.

It is worth mentioning at this point that the polytope Q_c depends only on the quantifier rank k of φ and the number of free variables of φ. The dependence on the formula φ itself only manifests in the choice of the face Q_φ of Q_c that projects to $P_\varphi(G)$. ◀

▶ **Corollary 8.** *The extension complexity of the convex hull of all satisfying assignments of a given MSO$_2$ formula φ on a given graph G of bounded treewidth is linear in the size of the graph G.*

5 Efficient Construction of the MSO Polytope

In the previous section we have proven that $P_\varphi(G)$ *has* a compact extended formulation but our definition of feasible tuples and the indicator functions μ and ρ_φ did not explicitly provide a way how to actually *obtain* it efficiently. That is what we do in this section. We also briefly mention some implications of our results for optimization versions of Courcelle's theorem.

As in the previous section we assume that we are given a graph G of treewidth τ and an MSO formula φ with m free variables and quantifier rank k. We start by constructing a nice tree decomposition T of G of treewidth τ in linear time.

Let $\mathcal{C}$ denote the set of equivalence classes of $\equiv_k^{MSO}$. Because $\mathcal{C}$ is finite and its size is independent of the size of G (Theorem 2), for each class $\alpha \in \mathcal{C}$, there exists an $[m]$-colored τ-boundaried graph $(G^\alpha, \vec{X}^\alpha, \vec{p^\alpha})$ of type α whose size is upper-bounded by a function of k, m and τ. For each $\alpha \in \mathcal{C}$, we fix one such graph, denote it by $W(\alpha)$ and call it the *witness of* α. Let $\mathcal{W} = \{W(\alpha) \mid \alpha \in \mathcal{C}\}$. The witnesses make it possible to compute the indicator function ρ_φ: for every $\alpha \in \mathcal{C}$, we set $\rho_\varphi(\alpha) = 1$ if and only if $W(\alpha) \models \varphi$, and we set $\rho_\varphi(\alpha) = 0$ otherwise.

▶ **Lemma 9** (implicitly in [16] in the proof of Theorem 4.6 and Corollary 4.7)**.** *The set $\mathcal{W}$ and the indicator function ρ_φ can be computed in time $f(k, m, \tau)$, for some computable function f.*

It will be important to have an efficient algorithmic test for MSO$[k, \tau]$-elementary equivalence. This can be done using the Ehrenfeucht-Fraïssé games:

▶ **Lemma 10** (Theorem 7.7 [30])**.** *Given two $[m]$-colored τ-boundaried graph $G_1^{[m],\tau}$ and $G_2^{[m],\tau}$, it can be decided in time $f(m, k, \tau, |G_1|, |G_2|)$ whether $G_1^{[m],\tau} \equiv_k^{MSO} G_2^{[m],\tau}$, for some computable function f.*

▶ **Corollary 11.** *Recognizing the type of an $[m]$-colored τ-boundaried graph $G^{[m],\tau}$ can be done in time $f(m, k, \tau, |G|)$, for some computable function f.*

Now we describe a linear time construction of the sets of feasible types, pairs and triples of types $\mathcal{F}(b)$, $\mathcal{F}_p(b)$ and $\mathcal{F}_t(b)$ for all relevant nodes b in T. In the initialization phase, we construct the set $\mathcal{W}$ and the indicator function ρ_φ using the algorithm from Lemma 9. The rest of the construction is inductive, starting in the leaves of T and advancing in a bottom up fashion towards the root of T. The idea is to always replace a possibly *large* graph $G_a^{[m],\tau}$ of type α by the *small* witness $W(\alpha)$ when computing the set of feasible types for the parent of the node a.

Leaf node. For every leaf node $a \in V(T)$ we set $\mathcal{F}(a) = \{\alpha_1\}$. Obviously, this corresponds to the definition in Section 3.

Introduce node. Assume that $b \in V(T)$ is an introduce node with a child $a \in V(T)$ for which $\mathcal{F}(a)$ has already been computed, and $v \in V(G)$ is the new vertex. For every $\alpha \in \mathcal{F}(a)$, we first produce a τ'-boundaried graph $H^{\tau'} = (H^\alpha, \vec{q})$ from $W(\alpha) = (G^\alpha, \vec{X^\alpha}, \vec{p^\alpha})$ as follows: let $\tau' = |\vec{p^\alpha}| + 1$ and H^α be obtained from G^α by attaching to it a new vertex in the same way as v is attached to G_a. The boundary $\vec{q}$ is obtained from the boundary $\vec{p^\alpha}$ by inserting in it the new vertex at the same position that v has in the boundary of $(G_a, \sigma(B(a)))$. For every subset $I \subseteq \{1, \ldots, m\}$ we construct an $[m]$-coloring $\vec{Y}^{\alpha,I}$ from $\vec{X^\alpha}$ by setting $Y_i^{\alpha,I} = X_i^\alpha \cup \{v\}$, for every $i \in I$, and $Y_i^{\alpha,I} = X_i^\alpha$, for every $i \notin I$. Each of these $[m]$-colorings $\vec{Y}^{\alpha,I}$ is used to produce an $[m]$-colored τ'-boundaried graph $(H^\alpha, \vec{Y}^{\alpha,I}, \vec{q})$ and the types of all these $[m]$-colored τ'-boundaried graphs are added to the set $\mathcal{F}(b)$ of feasible types of b, and, similarly, the pairs (α, β) where β is a feasible type of some of the $[m]$-colored τ'-boundaried graph $(H^\alpha, \vec{Y}^{\alpha,I}, \vec{q})$, are added to the set $\mathcal{F}_p(b)$ of all feasible pairs of types of b. The correctness of the construction of the sets $\mathcal{F}(b)$ and $\mathcal{F}_p(b)$ for the node b of T follows from Lemma 4.

Forget node. Assume that $b \in V(T)$ is a forget node with a child $a \in V(T)$ for which $\mathcal{F}(a)$ has already been computed and that the d-th vertex of the boundary $\sigma(B(a))$ is the vertex being forgotten. We proceed in a similar way as in the case of the introduce node. For every $\alpha \in \mathcal{F}(a)$ we produce an $[m]$-colored τ'-boundaried graph $(H^\alpha, \vec{Y}^\alpha, \vec{q})$ from $W(\alpha) = (G^\alpha, \vec{X^\alpha}, \vec{p^\alpha})$ as follows: let $\tau' = |\vec{p^\alpha}| - 1$, $H^\alpha = G^\alpha$, $\vec{Y}^\alpha = \vec{X^\alpha}$ and $\vec{q} = (p_1, \ldots, p_{d-1}, p_{d+1}, \ldots, p_{\tau'+1})$. For every $\alpha \in \mathcal{F}(a)$, the type β of the constructed graph is added to $\mathcal{F}(b)$, and, similarly, the pairs (α, β) are added to $\mathcal{F}_p(b)$. The correctness of the construction of the sets $\mathcal{F}(b)$ and $\mathcal{F}_p(b)$ for the node b of T follows from Lemma 5.

Join node. Assume that $c \in V(T)$ is a join node with children $a, b \in V(T)$ for which $\mathcal{F}(a)$ and $\mathcal{F}(b)$ have already been computed. For every pair of compatible types $\alpha \in \mathcal{F}(a)$ and $\beta \in \mathcal{F}(b)$, we add the type γ of $W(\alpha) \oplus W(\beta)$ to $\mathcal{F}(c)$, and the triple (α, β, γ) to $\mathcal{F}_t(c)$. The correctness of the construction of the sets $\mathcal{F}(c)$ and $\mathcal{F}_t(c)$ for the node b of T follows from Lemma 3.

It remains to construct the indicator function μ. We do it during the construction of the sets of feasible types as follows. We initialize μ to zero. Then, every time we process a node b in T and we find a new feasible type β of b, for every $v \in B(b)$ and for every i for which d-th vertex in the boundary of $W(\beta) = (G^\beta, \vec{X}, \vec{p})$ belongs to X_i, we set $\mu(\beta, v, i) = 1$ where d is the order of v in the boundary of $(G_b, \sigma(B(b)))$. The correctness follows from the definition of μ and the definition of feasible types.

Concerning the time complexity of the inductive construction, we observe, exploiting Corollary 11, that for every node b in T, the number of steps, the sizes of graphs that we worked with when dealing with the node b, and the time needed for each of the steps, depends on k, m and τ only. We summarize the main result of this section in the following theorem.

▶ **Theorem 12.** *Under the assumptions of Theorem 6, the polytope $P_\varphi(G)$ can be constructed in time $f'(|\varphi|, \tau) \cdot n$, for some computable function f'.*

5.1 Courcelle's Theorem and Optimization

It is worth noting that even though linear time *optimization* versions of Courcelle's theorem are known, our result provides a linear size LP for these problems out of the box. Together with a polynomial algorithm for solving linear programming we immediately get the following:

▶ **Theorem 13.** *Given a graph G on n vertices with treewidth τ, a formula $\varphi \in MSO$ with m free variables and real weights w_v^i, for every $v \in V(G)$ and $i \in \{1, \dots, m\}$, the problem*

$$opt\left\{ \sum_{v \in V(G)} \sum_{i=1}^{m} w_v^i \cdot y_v^i \,\middle|\, y \text{ satisfies } \varphi \right\}$$

where opt is min *or* max*, is solvable in time polynomial in the input size.*

Acknowledgements. We thank the anonymous reviewers for pointing out existing work and shorter proof of the Glueing lemma, among various other improvements.

References

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

2 D. Avis and H. R. Tiwary. On the extension complexity of combinatorial polytopes. In *Proc. ICALP(1)*, pages 57–68, 2013.

3 D. Bienstock and G. Munoz. LP approximations to mixed-integer polynomial optimization problems. *ArXiv e-prints*, January 2015. arXiv:1501.00288.

4 H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proc. STOC*, pages 226–234, 1993.

5 H. L. Bodlaender. Treewidth: characterizations, applications, and computations. In *Proc. of WG*, volume 4271 of *LNCS*, pages 1–14. Springer, 2006.

6 G. Braun, S. Fiorini, S. Pokutta, and D. Steurer. Approximation limits of linear programs (beyond hierarchies). *Math. Oper. Res.*, 40(3):756–772, 2015.

7 G. Braun, R. Jain, T. Lee, and S. Pokutta. Information-theoretic approximations of the nonnegative rank. *Electronic Colloquium on Computational Complexity*, 20:158, 2013.

8 A. Buchanan and S. Butenko. Tight extended formulations for independent set, 2014. Available on Optimization Online. URL: http://www.optimization-online.org/DB_HTML/2014/09/4540.html.

9 M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

10 M. Conforti and K. Pashkovich. The projected faces property and polyhedral relations. *Mathematical Programming*, pages 1–12, 2015.

11 B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

12 B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. In *Proc. of WG*, volume 1517 of *LNCS*, pages 125–150. Springer, 1998.

13 B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1–2):49–82, 1 March 1993.

14 Y. Faenza, S. Fiorini, R. Grappe, and H. R. Tiwary. Extended formulations, nonnegative factorizations, and randomized com. protocols. *Math. Program.*, 153(1):75–94, 2015.

15 S. Fiorini, S. Massar, S. Pokutta, H. R. Tiwary, and Ronald de Wolf. Exponential lower bounds for polytopes in combinatorial optimization. *J. ACM*, 62(2):17, 2015.

16 G. Gottlob, R. Pichler, and F. Wei. Monadic datalog over finite structures with bounded treewidth. In *Proc. PODS*, pages 165–174, 2007.

17 B. Grünbaum. *Convex Polytopes.* Wiley Interscience Publ., London, 1967.

18 V. Kaibel. Extended formulations in combinatorial optimization. *Optima*, 85:2–7, 2011.

19 V. Kaibel and A. Loos. Branched polyhedral systems. In *Proc. IPCO*, volume 6080 of *LNCS*, pages 177–190. Springer, 2010.

20 V. Kaibel and K. Pashkovich. Constructing extended formulations from reflection relations. In *Proc. IPCO*, volume 6655 of *LNCS*, pages 287–300. Springer, 2011.

21 L. Kaiser, M. Lang, S. Leßenich, and Ch. Löding. A Unified Approach to Boundedness Properties in MSO. In *Proc. of CSL*, volume 41 of *LIPIcs*, pages 441–456, 2015.

22 T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

23 J. Kneis, A. Langer, and P. Rossmanith. Courcelle's theorem – A game-theoretic approach. *Discrete Optimization*, 8(4):568–594, 2011.

24 P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proc. PODS*, 1998.

25 P. Kolman and M. Koutecký. Extended formulation for CSP that is compact for instances of bounded treewidth. *Electr. J. Comb.*, 22(4):P4.30, 2015.

26 S. Kreutzer. Algorithmic meta-theorems. In *Proc. of IWPEC*, volume 5018 of *LNCS*, pages 10–12. Springer, 2008.

27 A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Computer Science Review*, 13-14:39–74, 2014.

28 M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

29 J. R. Lee, P. Raghavendra, and D. Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proc. STOC*, pages 567–576, 2015.

30 L. Libkin. *Elements of Finite Model Theory.* Springer, Berlin, 2004.

31 F. Margot. *Composition de polytopes combinatoires: une approche par projection.* PhD thesis, École polytechnique fédérale de Lausanne, 1994.

32 R. K. Martin, R. L. Rardin, and B. A. Campbell. Polyhedral characterization of discrete dynamic programming. *Oper. Res.*, 38(1):127–138, February 1990.

33 M. Sellmann. The polytope of tree-structured binary constraint satisfaction problems. In *Proc. CPAIOR*, volume 5015 of *LNCS*, pages 367–371. Springer, 2008.

34 M. Sellmann, L. Mercier, and D. H. Leventhal. The linear programming polytope of binary constraint problems with bounded tree-width. In *Proc. CPAIOR*, volume 4510 of *LNCS*, pages 275–287. Springer, 2007.

35 F. Vanderbeck and L. A. Wolsey. Reformulation and decomposition of integer programs. In *50 Years of Integer Programming 1958-2008*, pages 431–502. Springer, 2010.

36 L. A. Wolsey. Using extended formulations in practice. *Optima*, 85:7–9, 2011.

37 M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. Syst. Sci.*, 43(3):441–466, 1991.

38 G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1995.