Context-Free Graph Properties via Definable Decompositions

Michael Elberfeld

RWTH Aachen University, Aachen, Germany
elberfeld@informatik.rwth-aachen.de

Abstract

Monadic-second order logic (MSO-logic) is successfully applied in both language theory and algorithm design. In the former, properties definable by MSO-formulas are exactly the regular properties on many structures like, most prominently, strings. In the latter, solving a problem for structures of bounded tree width is routinely done by defining it in terms of an MSO-formula and applying general formula-evaluation procedures like Courcelle's. The present paper furthers the study of second-order logics with close connections to language theory and algorithm design beyond MSO-logic.

We introduce a logic that allows to expand a given structure with an existentially quantified tree decomposition of bounded width and test an MSO-definable property for the resulting expanded structure. It is proposed as a candidate for capturing the notion of "context-free graph properties" since it corresponds to the context-free languages on strings, has the same closure properties, and an alternative definition similar to the one of Chomsky and Schützenberger for context-free languages. Besides studying its language-theoretic aspects, we consider its expressive power as well as the algorithmics of its satisfiability and evaluation problems.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases finite model theory, monadic second-order logic, tree decomposition, context-free languages, expressive power

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.17

1 Introduction

The properties that are definable by formulas from monadic second-order logic (MSO-logic) correspond to the notion of regularity for many classes of structures. Most prominently, the MSO-definable properties of strings correspond to the regular languages [3, 8, 19] and this also holds for regular languages of trees [18, 6]. Moreover, a recent work [2] generalizes this connection to the MSO-definable properties of every class of structures with bounded tree width (that means, a class of structures that have tree decompositions whose width is bounded by a class-dependent constant).

Besides its application to study regular languages of various classes of tree-width-bounded structures, MSO-logic is frequently used to design algorithms for solving problems on these structures. Every MSO-definable property can be decided in polynomial time on every class of structures of bounded tree width, even if we are looking out for an algorithm that runs in linear time [4] or has a logarithmic memory footprint [7]. The usefulness of these results lies in the fact that, in order to prove that a certain concrete problem is decidable in, say, linear time on a class of structures of bounded tree width, we only need to show that it is MSO-definable. That means, writing down a defining MSO-formula for it. Then the linear time bound for solving the problem follows from the general result of [4].

Besides MSO-definable problems, there are many problems of both practical and theoretical interest that are not MSO-definable. In order to define them, we need second-order formulas, which have the ability to establish new relations. Prominent examples of such properties are the context-free languages of strings. Since there are context-free languages that are not regular, they are not definable in terms of MSO-formulas. Lautemann, Schwentick, and Thérien [14] addressed this issue by presenting a logic for the context-free languages of strings. It extends MSO-logic with the ability to establish a binary relation, called a *matching* relation, that puts a string of well-formed nested brackets on top of a given string. While this logic generalizes the connection between formal languages and predicate logics from regular languages to context-free languages, it does not apply to more general structures like trees and graphs of bounded tree width. In particular, it does not provide us with a logic that helps to further the application of logics as a descriptive tool in algorithm design. Motivated by the comment of Lautemann et al. that "We are convinced that a more general study of these logics will prove worthwhile in the context of general finite structures, instead of strings", the present paper remedies this situation by presenting an extension of MSO-logic that is equivalent to the context-free languages of strings, and also generalizes to every class of structures of bounded tree width.

Contributions. We introduce a logic that generalizes MSO on bounded tree width graphs by (1) allowing to existentially guess a tree decomposition of some bounded width w for a given graph, and (2) test an MSO-definable property for a structure that encodes both the given graph and the guessed tree decomposition. That means, the logic still defines a property of graphs, but the defined property is based on cycling through candidate tree decompositions of them. We call the newly proposed logic MSO-*logic with quantified tree decompositions* (DMSO-logic). The main purpose of the present paper is to propose this logic as a candidate for capturing the notion of "context-free properties of graphs" with strong connections to both formal languages and algorithm design.

In order to tighten the connection of DMSO-logic with formal languages, we prove that its definable properties of strings are exactly the context-free languages while already a slight modification of the decompositions that we use leads to a logic that also defines properties that are not context-free. Moreover, we study the closure of DMSO-definable properties under set operations and show that they are similar to the closure properties of the context-free languages. Our logic also allows for an alternative characterization similar to the Theorem of Chomsky and Schützenberger, which characterizes context-free languages in terms of words containing well-nested brackets and a regular property of these words.

For establishing a connection between DMSO-logic and algorithm design, we study the complexity of its satisfiability and formula-evaluation problems. Its satisfiability problem turns out to be decidable. Deciding whether a given structure is a model of a DMSO-formula turns out to be polynomial-time computable for every fixed formula. In order to obtain the polynomial-time upper bound, we need to work with MSO-properties of candidate tree decompositions without constructing them explicitly. We work with logical types and develop an inductive approach that computes the types of substructures and their candidate decompositions. In order to understand the range of applications of our logic, we also study its expressive power. Since MSO-logic on structures of bounded tree width is well studied, we concentrate on the main distinguishing feature of DMSO-logic: the quantified tree decompositions. We study how a varying width of the quantified tree decomposition influences the expressive power of the logic. Again, using logical types to reasoning about candidate decompositions plays an important role for proving the expressivity results.

Organization. After introducing the necessary background on MSO-logic in Section 2, we define DMSO-logic in Section 3. Sections 4, 5, and 6 present results related to the language-theoretic, model-theoretic, and algorithmic aspects of DMSO-logic, respectively. Section 7 concludes with a discussion of the presented results and directions for future work.

2 Background on Monadic Second-Order Logic

A *vocabulary* τ is a finite set of *relational symbols* where an *arity* $\mathrm{ar}(R) \geq 1$ is assigned to each symbol $R \in \tau$. A *structure* A over τ (also called τ-structure) consists of a finite set $U(A)$, its *universe*, and a *relation* $R(A) \subseteq U(A)^{\mathrm{ar}(R)}$ for every $R \in \tau$. The *tuples* of A are all $(a_1, \ldots, a_{\mathrm{ar}(R)}) \in R(A)$ for some $R \in \tau$. The substructure of A that is *induced* by a set of elements $U' \subseteq U(A)$ is denoted by $A[U']$; it has universe U' and consists of all tuples from A whose elements are in U'. We view a *graph* G as a structures over the vocabulary $\tau_{\text{graphs}} := \{E^2\}$; in addition to the above notation we denote its universe also by $V(G)$ and call its elements *vertices*, and call the tuples in $E(G)$ *edges*. A graph is *undirected* if $E(G)$ is symmetric.

To define the *syntax* of *second-order logic* (SO-logic), we use *element variables* (also called *first-order variables*) x_i for $i \in \mathbb{N}$ and *relation variables* (also called *second-order variables*) X_i for $i \in \mathbb{N}$ that have an arity $\mathrm{ar}(X_i) \geq 1$. *Formulas of* SO*-logic* (SO*-formulas*) over a vocabulary τ are inductively defined as follows: *Atomic* formulas are all $x_i = x_j$, $(x_{i_1}, \ldots, x_{i_r}) \in X_j$ with $r = \mathrm{ar}(X_j)$, and $(x_{i_1}, \ldots, x_{i_r}) \in R$ for $R \in \tau$ with $r = \mathrm{ar}(R)$. Given SO-formulas φ_1 and φ_2, *composed* formulas are build from *element quantifiers* via $\exists x_i\,(\varphi_1)$ and *relation quantifiers* via $\exists X_i\,(\varphi_1)$ as well as *Boolean connectives* $\varphi_1 \wedge \varphi_2$ and $\neg\,(\varphi_1)$. The set of *free variables* of an SO-formula φ, denoted by $\mathrm{free}(\varphi)$, contains the variables of φ that are not used as part of a quantification. By renaming a formula's variables, we can always assume $\mathrm{free}(\varphi) = \{x_1, \ldots, x_k, X_1, \ldots, X_\ell\}$ for some $k, \ell \in \mathbb{N}$; we write $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ to indicate that the free variables of φ are exactly x_1 to x_k and X_1 to X_ℓ. An SO*-sentence* is an SO-formula without free variables; we use the term sentence in a similar way for other formulas from other logics as well. Regarding the *semantics* of SO-logic, we write $A \models \varphi(a_1, \ldots, a_k, A_1, \ldots, A_\ell)$ to indicate that an SO-formula $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ evaluates to true for a structure A over the same vocabulary and an *assignment* $(a_1, \ldots, a_k, A_1, \ldots, A_\ell) \in U^k \times \mathrm{pow}(U^{\mathrm{ar}(X_1)}) \times \cdots \times \mathrm{pow}(U^{\mathrm{ar}(X_\ell)})$ to φ's free variables, where $\mathrm{pow}(B)$ denotes the power set of a set B.

Monadic second-order logic (MSO-logic) is defined by taking all SO-formulas without second-order variables (neither free nor bound) of arity 2 and higher along with the semantics of SO-logic; we call the second-order variables just *set variables* when working with MSO-logic. *Modulo-counting monadic second-order logic* (CMSO-logic) is an extension of MSO-logic that also allows to use *modulo-counting atomic formulas* $\mathrm{MOD}_m(X)$ for every $m \in \mathbb{N}$ and set variable X. We write $A \models \mathrm{MOD}_m(A_1)$ for a set $A_1 \subseteq U(A)$ exactly if the number of elements in A_1 is a multiple of m.

A set of τ-structures $\mathcal{P}$ that is closed under taking isomorphic structures is also called a *property* of τ-structures. It is SO*-definable* if there is an SO-formula φ over τ with $A \models \varphi$ for τ-structures $A \in \mathcal{P}$ and $A \not\models \varphi$ for τ-structures $A \notin \mathcal{P}$. Given a second set of structures $\mathcal{C}$, we say that $\mathcal{P}$ is SO*-definable on* $\mathcal{C}$ if $\mathcal{P} \cap \mathcal{C}$ is SO-definable. Note that this definition does not consider membership in $\mathcal{C}$ as a promise, which all given structures meet by assumption. We use the notion of *definable* and *definable on* also for other logics like, for example, MSO-logic and CMSO-logic.

3 Monadic Second-Order Logic with Quantified Tree Decompositions

The present section introduces a logic that we propose as an accessible framework for capturing the notion of context-free properties of graphs and, more general, structures. It is based on guessing a constant-width tree decomposition for a given structure and checking a property for the decomposition and the structure that is MSO-definable. In order to introduce the logic, we review the notion of tree decompositions and how they can be encoded as part of structures in Section 3.1. Based on this concept, we define our logic in Section 3.2.

3.1 Structures Expanded by Tree Decompositions

A *directed tree* T is a tree whose edges are oriented away from a unique *root* $\mathrm{r}(T) \in V(T)$; we call is vertices *nodes* and edges *arcs*. A *tree decomposition* $D = (T, \beta)$ of a structure A is a directed tree T together with a labeling function $\beta\colon V(T) \to \mathrm{pow}(U(A))$ satisfying the below connectedness, covering, and strictness conditions. For every $t \in V(T)$, the set $\beta(t)$ is the *bag* of t. The *width* of D is $\max_{t \in V(T)} |\beta(t)| - 1$.

- *(Connectedness)* For every element $a \in U(A)$, the induced substructure $T\big[\{t \in V(T) \mid a \in \beta(t)\}\big]$ is a nonempty directed tree.
- *(Covering)* For every tuple $(a_1, \dots, a_r)$ of A, there is a $t \in V(T)$ with $\{a_1, \dots, a_r\} \subseteq \beta(t)$.

In order to state the strictness condition, we introduce some terminology partly adapted from [11]. The root of $D = (T, \beta)$ is $\mathrm{r}(D) := \mathrm{r}(T)$. The *separator* of t is $\sigma(t) := \beta(u) \cap \beta(t)$ if it has a parent u and $\sigma(t) := \emptyset$ if $t = \mathrm{r}(T)$. For every $t \in V(T)$, T_t is the subtree of T rooted at t. Then $\gamma(t)$, called the *cone* of t, is the union of all bags $\beta(u)$ for $u \in V(T_t)$. The *component* of t is $\alpha(t) := \gamma(t) \setminus \sigma(t)$. Besides the two conditions above, we require the following third condition that is commonly met by the tree decompositions used in the literature, but normally not made explicit as part of the definition.

- *(Strictness)* For every $(u, t) \in E(T)$, we have $\gamma(u) \supsetneq \gamma(t)$ or $\alpha(u) \supsetneq \alpha(t)$.

The *Gaifman graph* $G(A)$ of a structure A has vertex set $V(G(A)) := U(A)$ and there is an edge $(u, v) \in E(G(A))$ exactly if u and v are part of a common tuple t of A; thus, $G(A)$ is undirected. Instead of directly working with tree decompositions for structures, we sometimes use the following folklore fact (see [10] for a formal proof) to consider tree decompositions for their Gaifman graphs.

▶ **Fact 1.** *Every tree decomposition of a structure A is also a tree decomposition of its Gaifman graph $G(A)$, and vice versa.*

Let A be a structure over some vocabulary τ and $D = (T, \beta)$ a tree decomposition for it. In order to have a single structure that encodes both A and D, we define the *expansion* (A, D) of A by D as follows: Its universe is the disjoint union of $U(A)$ and $V(T)$. It contains all relations of A along with the unary relation $\textsc{nodes} := V(T)$, which is used to distinguish between nodes from the tree decomposition and elements from the structure, the binary relation $\textsc{arcs} := E(T)$, to encode the directed tree underlying the tree decomposition, and $\textsc{bags} := \{(t, a) \in V(T) \times U(A) \mid a \in \beta(t)\}$, to encode the assignment of elements to the bags. Thus, (A, D) is a structure over the vocabulary $\tau^\star := \tau \cup \{\textsc{nodes}, \textsc{arcs}, \textsc{bags}\}$ where we assume that the relation symbols NODES, ARCS, and BAGS are not part of τ. If we move from an expansion (A, D) back to A, we speak of *reducing* (A, D). Courcelle and Engelfriet [5, Example 5.2] discuss other ways of how to encode a graph along with a tree decomposition as a single relational structure. The definition in the present paper follows the work of Adler et al. [1].

The tree width $\mathrm{tw}(A)$ of a structure A is the minimum width among all its tree decompositions. The following lemma implies that the tree width of a structure only grows by a constant additive factor if we expand it with a tree decomposition of minimum possible width. The lemma without a proof can also be found in [1].

▶ **Lemma 2.** *Let A be a structure with a tree decomposition $D = (T, \beta)$ of width w. Then the tree width of (A, D) is at most $w + 2$.*

Proof. We show how to build a width-$(w + 2)$ tree decomposition $D' = (T', \beta')$ for (A, D), which proves the lemma. As the underlying tree T', we use a copy of T and rename each node $t \in V(T)$ into t'. We set $\beta'(t') := \{u, t\} \cup \beta(t)$ if t has a parent u in T and $\beta'(t') := \{t\} \cup \beta(t)$ if $t = \mathrm{r}(T)$. Since D is already a tree decomposition for A, this construction satisfies the connectedness condition for all elements from $U(A)$ and the cover condition for all tuples from A. A node u from $V(T)$ only appears in the bag of u' and the bag of each t' where t is a child of u in T. Thus, the connectedness condition holds for the elements from $V(T)$. Moreover, the construction puts every edge $(u, t) \in E(T)$ into the bag $\beta'(t')$. Strictness is inherited from D and the width bound $w + 2$ follows from the construction. ◀

3.2 Quantified Tree Decompositions

For the definition of the logic, we use tree decompositions $D = (T, \beta)$ for structures A that are normalized in the sense of satisfying the following condition.

- (Normal) For every $t \in V(T)$, $G[\alpha(t)]$ is connected, where $G = G(A)$.

This condition, which is commonly met by tree decompositions that are constructed via recursive separator-based approaches, is a necessary ingredient for proving the equivalence to the context-free languages of our logic on strings. This can be seen by comparing the tree decompositions used in Theorem 6, which are normal, with the tree decompositions used in Lemma 10, which are not normal in general.

Compared to MSO-logic, the following logic allows to establish new relations beyond just sets, but the relations encode a tree decomposition of bounded width. Definition 4 uses the MSO-sentences from the following proposition, which can be written down by formulating the above conditions for tree decompositions in terms of subformulas.

▶ **Proposition 3.** *For every vocabulary τ and $w \in \mathbb{N}$, there is an* MSO*-sentence $\varphi_{\text{tw-}w\text{-dec}}$ over $\tau^\star$ that defines the τ-structures that are expanded by normal tree decompositions of width at most w.*

▶ **Definition 4.** We define the syntax and semantics of MSO*-logic with quantified tree decompositions* (DMSO-logic) as follows:

- (Syntax) A DMSO-formula over some vocabulary τ has the form $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ where $\varphi_{\text{tw-}w\text{-dec}}$ is from Proposition 3 and φ is an MSO-sentence over $\tau^\star$.
- (Semantics) For a τ-structure A and a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$, we write $A \models \psi$ exactly if there is a tree decomposition D with $(A, D) \models \varphi_{\text{tw-}w\text{-dec}} \wedge \varphi$.

Similar to how CMSO-logic extends MSO-logic, we extend DMSO-logic to a modulo-counting variant by allowing φ to be a CMSO-sentence instead of just an MSO-sentence. The resulting logic is called CMSO*-logic with quantified tree decompositions* (DCMSO-logic).

While DMSO-logic implicitly guesses a tree decomposition, which includes establishing new elements, it can, equivalently, be defined as a syntactic fragment of SO-logic. In this case, we use higher-ary tuples to encode elements of the tree decomposition. In order to have

a clean presentation, we use the definition given above, but keep in mind that DMSO-logic can be defined as a syntactic fragment of SO-logic.

The property defined by the MSO-subformula φ in Definition 4 can, possibly, depend on the shape of the tree decomposition that is guessed by the decomposition quantifier. A variant of this definition would be to allow an existentially guessed tree decomposition, but restrict the property of φ to be invariant with respect to the chosen tree decomposition. In this case, the guessed tree decomposition supports the formula φ in defining a property of the given structure rather than φ defining a property of the structure and the tree decomposition. Interestingly, this invariant use of a width-bounded tree decomposition corresponds to the notion of recognizable properties of graphs of bounded tree width as studied in [2].

4 Language-Theoretic Aspects

The present section shows that the DMSO-definable properties of strings are exactly the context-free languages (Section 4.1). Moreover, we show that modifying DMSO-logic in terms of using decompositions that are not normal leads to a more expressive logic (Section 4.2).

4.1 DMSO-Definability on Strings Equals the Context-Free Languages

In order to move back and forth between formal languages and logic, we encode strings by structures that have the shape of a path with unary relations to encode symbols. For an alphabet $\Sigma = \{S_1, \dots, S_m\}$, we consider the vocabulary $\tau_\Sigma := \{S_1^1, \dots, S_m^1\}$. A Σ-*path* is a structure P over the vocabulary $\tau_{\Sigma\text{-paths}} := \text{SUCC}^2 \cup \tau_\Sigma$ where $\text{SUCC}(P)$ is the edge relation of a directed path and the unary relations $S_1(P)$ to $S_m(P)$ partition $U(P)$. Strings over an alphabet Σ translate into Σ-paths and back if we use $\text{SUCC}(P)$ to encode the concatenation and the $S_i(P)$ to encode symbols. Given a language $L \subseteq \Sigma^*$ for some alphabet Σ, we denote by $\mathcal{P}(L)$ the set of Σ-paths that encode strings from L.

Based on this notation, we are able to state the equivalence of the regular languages and the MSO-definable string properties of Büchi, Elgot, and Trakhtenbrot [3, 8, 19].

▶ **Fact 5.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is regular if, and only if, $\mathcal{P}(L)$ is* MSO*-definable.*

Similar to this fact, during the course of the present section, we prove the following connection between the DMSO-definable properties of strings and the context-free languages.

▶ **Theorem 6.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is context-free if, and only if, $\mathcal{P}(L)$ is* DMSO*-definable.*

The theorem follows from Lemmas 8 and 9, which show how to move from context-free languages to DMSO-definable string properties and back, respectively. Moreover, the lemmas imply that, in order to capture the context-free languages, we only need tree decompositions of width at most 3. By Lemma 9, we do not leave the context-free languages if we use tree decompositions with a width bound that is larger than 3, but we can always go down to width 3 by applying Lemma 8 to the result of Lemma 9.

Besides using a context-free grammar or a pushdown automaton, a context-free language can be defined as the set of strings that are the yields of trees of a regular tree language; meaning that a string is generated by taking a binary tree with distinguished left and right children and reading the symbols at the leaves from left to right with respect to the ordering induced by the inner nodes of the tree. Since regular tree languages are exactly the MSO-definable properties of trees, this gives rise to a descriptive characterization of the

context-free languages in terms of trees and MSO-logic. We review this connection between context-free languages and MSO-definable trees in order to prove Lemmas 8 and 9.

A Σ-*tree* is a structure T over the vocabulary $\tau_{\Sigma\text{-trees}} := \{\text{LEFT}^2, \text{RIGHT}^2\} \cup \tau_\Sigma$ where relations $\text{LEFT}(T)$ and $\text{RIGHT}(T)$ encode a directed binary tree whose edges are partitioned into *left successors* $\text{LEFT}(T)$ and *right successors* $\text{RIGHT}(T)$. The unary relations partition the nodes of the tree (in fact, for the application to a characterization of the context-free languages, we only need that they partition the leaves of the tree). A Σ-tree *yields* a Σ-path as follows: Starting at T's root $\text{r}(T)$, we traverse the tree based on the depth-first search that walks to left successors with higher priority and to right successors with lower priority. This walk induces a total ordering on the leaves of the tree. In turn, it gives rise to a Σ-path P whose nodes are the leaves of T; this path is the *yield* of T.

We get the following fact from combining [16] with [18, 6] (see also its application in [14]).

▶ **Fact 7.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is context-free if, and only if, there is an* MSO*-definable property of Σ-trees $\mathcal{T}$ that yield exactly the Σ-paths $\mathcal{P}(L)$.*

▶ **Lemma 8.** *For every* MSO*-sentence φ over $\tau_{\Sigma\text{-trees}}$, there is an* MSO*-sentence $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$ satisfying the following for every Σ-path P: there is a Σ-tree T that yields P with $T \models \varphi$ if, and only if, there is a width-3 normal tree decomposition D for P with $(P, D) \models \varphi^\star$.*

Proof. For the proof we use a transformation from Σ-trees that yield paths to tree decompositions for paths. Based on this transformation, we turn a formula φ over $\tau_{\Sigma\text{-trees}}$ into a formula $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$ that meets the requirements of the lemma.

We start to describe how a Σ-tree T that yields a path P is turned into a tree decomposition $D = (T', \beta)$ for P. The tree underlying the decomposition is the tree T without distinguishing left and right successors where we rename every node $t \in V(T)$ into $t' \in V(T')$. That means $V(T') := V(T)$ and $E(T') := \text{LEFT}(T) \cup \text{RIGHT}(T)$ up to renaming nodes. We view the nodes of T' as being partitioned into the set of leaves, which are exactly the elements of P, and the inner nodes. The bags $\beta\colon V(T') \to \text{pow}(U(P))$ are defined as follows: For every $t' \in V(T')$, let P_t be the path that is the yield of the subtree T_t rooted at t. If t' is a leaf, then set $\beta(t') := U(P_t) = \{t\}$. If t' is an inner node, let l' and r' be its left child and right child, respectively, in T. Then $\beta(t')$ contains 4 elements of P_t: the left-most and right-most nodes of both P_l and P_r. Then $D = (T', \beta)$ is a tree decomposition for P due to the following reasons: Every unary tuple of P is already covered by the bags of the leaves. For a binary tuple $(p_1, p_2) \in \text{SUCC}(P)$, let l be the highest node in T, such that the right-most node of P_l is p_1, and let r be the highest node in the tree, such that the left-most node of P_r is p_2. The nodes l and r are children of a common parent node t and $\{p_1, p_2\} \subseteq \beta(t')$ holds. The connectedness condition is satisfied due to the following reason: A node of P is part of a leaf bag, stays in the bags above it while it is the left-most or right-most node of the path that is the yield of a subtree, stays in the bags while the two subtrees are merged, and is deleted from the bag afterwards. In particular, the bags that contain an element from P make up a path in T'. The width is 3 since bags have size at most 4. Moreover, the decomposition is strict since the cones of the nodes cover ever larger subpaths due the construction. Every component $\alpha(t)$ for $t \in V(T')$ is the vertex set of a subpath of P and, thus, induces a substructure with a connected Gaifman graph; that means, D is normal.

We want $\varphi^\star$ to (1) define a tree decomposition of the shape described above, and, based on this, (2) mimic the behavior of φ on T. We start with the first requirement. A first part of the formula $\varphi^\star$ ensures that the tree underlying the decomposition is binary, leaves have singleton bags that partition the set of elements of P, and inner nodes have bags of size 4 and are constructed in the way described above—their cones are subpaths and they split

into two subpaths that are, in turn, the cones of the two child nodes. A second part of the formula accesses the tree T' underlying D and reconstructs the partition of children into left successors and right successors. This can be done by also using the successor relation of P. In total, this reconstructs all the information we need to simulate the behavior of φ on T.

Correctness follows from two observations: First, every tree T whose yield is P can be transformed into a tree decomposition D of the form described above and $T \models \varphi$ implies $(P, D) \models \varphi^\star$. Moreover, a tree decomposition D of the kind defined by the first part of $\varphi^\star$ can be turned back into a tree T that yields P, such that $(P, D) \models \varphi^\star$ implies $T \models \varphi$. ◀

▶ **Lemma 9.** *For every $w \in \mathbb{N}$ and* MSO*-sentence $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$, there is an* MSO*-sentence φ over $\tau_{\Sigma\text{-trees}}$ satisfying the following for every Σ-path P: There is a width-w normal tree decomposition D for P with $(P, D) \models \varphi^\star$ if, and only if, there is a Σ-tree T that yields P with $T \models \varphi$.*

Proof. We start to give a transformation of normal tree decompositions for paths into trees that yield paths. Then we show how this transformation can be used to prove the lemma.

Let $D = (T, \beta)$ be a width-w normal tree decomposition for P. Due to the normalization, we know that every component $\alpha(t)$ for $t \in V(T)$ is connected and, since P is a path, it is a connected subpath P_t of P. Due to the strictness condition, the subpaths get larger when moving from a child to its parent or the bag size grows. Consider a node t of T and its children c_1 to c_ℓ, and let P_t and P_1 to P_ℓ be the paths that are induced by the respective components. Distinct paths P_i and P_j for $i, j \in \{1, \ldots, \ell\}$ have distinct elements and, using the at most $w + 1$ elements from $\beta(t)$, they are connected to form a path P_t. In particular, this means that ℓ is upper bounded in terms of w. To turn D into a tree T' that yields P we start with the tree T and insert additional nodes and edges to make it binary. First of all, we call the nodes of T *decomposition nodes*. Every decomposition node t is replaced by a chain of $|\beta(t) \setminus \sigma(t)| + \ell$ nodes; the chain is based on using binary relations from $\textsc{right}(T')$. The beginning of the chain is connected to the parent of t. Using binary relations from $\textsc{left}(T')$, we add edges to leaf nodes for the elements from $\beta(t) \setminus \sigma(t)$ and edges to the child nodes c_1 to c_ℓ to this chain. The order of adding edges is based on the order of the nodes and subpaths on the path P. We call the newly established leaves *element nodes*. The resulting binary ordered tree T' yields P via traversing the leaves based on the ordering that is induced by the tree.

For turning $\varphi^\star$ into φ, we first use subformulas that single out the decomposition and element nodes, respectively. This enables us to reconstruct the tree decomposition D from T'. Thus, we have access to all of (P, D) and define the property defined by $\varphi^\star$.

Correctness follows from the fact that φ over Σ-trees defines exactly the Σ-trees that encode decompositions as described above. Thus, we can move from Σ-trees to tree decompositions and back while maintaining the answer to the model relation for $\varphi^\star$ and φ. ◀

4.2 Beyond Context-Free Languages via General Decompositions

The logic DMSO is based on normal tree decompositions instead of general ones. The main reason behind this is based on the fact that, using general tree decompositions of bounded width, it is possible for an MSO-formula to define properties of strings that are not context-free. This is formally stated by the following lemma.

▶ **Lemma 10.** *Let $\Sigma = \{a, b, c\}$ and $L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$. There is an* MSO*-formula φ satisfying the following for every Σ-path P: we have $P \in \mathcal{P}(L)$ if, and only if, there is a width-3 tree decomposition D for P with $(P, D) \models \varphi$.*

Proof. The tree decompositions that are used to prove the lemma have the shape of a path. The root bag contains the left-most a-labeled node, the right-most b-labeled node and the left-most c-labeled node. Starting from these nodes, the a-labeled, b-labeled, and c-labeled subpaths are processed by visiting one node after the other in an alternating fashion. This means that the a-labeled nodes are processed from left to right, the b-labeled nodes are processed from right to left, and the c-labeled nodes are processed from left to right. In order to move from one node to the other, we add a single node to the current bag and, in the next step, this node replaces the prior one with the same label. The bags of the resulting decomposition have size at most 4. The properties of the decomposition, which are described above, are MSO-definable and the alternating replacement of nodes can be utilized to test whether the given path encodes a string $a^n b^n c^n$ for some $n \in \mathbb{N}$. ◀

The language L used in the previous lemma is not context-free with respect to the classical definition of context-free languages in terms of context-free grammars [13]. Nevertheless, other definitions of context-freeness in the context of graph grammars turn L into a context-free language [5].

5 Model-Theoretic Aspects

The logic DMSO on strings defines exactly the context-free languages as shown in the previous section. In the present section, we further the understanding of its expressive power for general structures. We start to show that its closure properties are similar to the ones of the context-free languages (Section 5.1). A DMSO-formula consists of a decomposition quantifier along with an MSO-formula that has access to both the structures and a tree decomposition for it. Since the expressive power of MSO-logic is well-understood (see, for example, [15] and [5]), we concentrate on the key aspect that DMSO-logic adds to MSO-logic: how the decomposition quantifier in conjunction with the built-in width bound influences the expressive power. We show that, already on graphs of tree width 1, increasing the width bound results in a higher expressive power (Section 5.2).

5.1 Closure of Definable Properties Under Set Operations

In the following, we develop properties that support DMSO-logic in being a reasonable logic-based generalization of the notion of context-freeness from strings to general structures. DMSO-logic has closure properties similar to the context-free languages and they exist for similar reasons. Moreover, it has an alternative characterization that is similar to the one of Chomsky and Schützenberger for the context-free languages.

▶ **Lemma 11.** *There is a vocabulary τ and* DMSO*-definable properties of τ-structure $\mathcal{P}$, $\mathcal{P}_1$, and $\mathcal{P}_2$, such that (complement) $\overline{\mathcal{P}}$ and (intersection) $\mathcal{P}_1 \cap \mathcal{P}_2$ are not* DMSO*-definable.*

Proof. To prove the lemma, we use $\tau_{\Sigma\text{-paths}}$ with $\Sigma = \{a, b, c\}$ and sets of Σ-paths $\mathcal{P} := \mathcal{P}(\{a^\ell b^m b^n \mid \ell \neq m \text{ or } m \neq n\})$, $\mathcal{P}_1 := \mathcal{P}(\{a^m b^n c^n \in \Sigma^* \mid m, n \in \mathbb{N}\})$, and $\mathcal{P}_2 := \mathcal{P}(\{a^m b^m c^n \in \Sigma^* \mid m, n \in \mathbb{N}\})$. The properties $\mathcal{P}$, $\mathcal{P}_1$, and $\mathcal{P}_2$ are DMSO-definable since they are based on context-free languages and we have Theorem 6. The complement of $\mathcal{P}$ restricted to Σ-paths is $\mathcal{P}' = \mathcal{P}(L)$ with language $L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$. Since L is not context-free (see, for example, Harrison's book [13] for a proof), Theorem 6 implies that $\overline{\mathcal{P}}$, the complement property of $\mathcal{P}$, is not DMSO-definable. Since $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathcal{P}(L)$, where L is the just defined language, which is not context-free, the non-closure with respect to intersection holds for the same reason. ◀

While standard closure properties of the context-free languages follow, for example, from the link to regular tree languages, which we used in Section 4, the closure properties of DMSO-logic follow from the syntax of their formulas.

▶ **Lemma 12.** *Let τ be a vocabulary, $\mathcal{P}_1$ and $\mathcal{P}_2$ DMSO-definable properties of τ-structures, and $\mathcal{P}$ an MSO-definable property of τ-structure. Then (union) $\mathcal{P}_1 \cup \mathcal{P}_2$ and (intersection with an MSO-property) $\mathcal{P}_1 \cap \mathcal{P}$ are DMSO-definable.*

Proof. Let $\psi_1 = \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi_1]$ and $\psi_2 = \exists D\,[\varphi_{\text{WIDTH-}w_2\text{-TD}} \wedge \varphi_2]$ be the DMSO-formulas that define $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, and φ be the MSO-formula that defines $\mathcal{P}$.

(union) Without loss of generality assume $w_1 \geq w_2$. To define the union $\mathcal{P}_1 \cup \mathcal{P}_2$, we use the formula $\psi' := \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi']$ with $\varphi' := \varphi_1 \vee [\varphi_{\text{WIDTH-}w_2\text{-TD}} \wedge \varphi_2]$. That means, the tree decomposition that we use for the inner part of the formula has width at most w_1 and satisfies φ_1 or it has a, possibly smaller, width w_2 and satisfies φ_2. This defines exactly the union of $\mathcal{P}_1$ and $\mathcal{P}_2$.

(intersection with an MSO-property) To define the intersection of $\mathcal{P}_1$ and $\mathcal{P}$, we use the formula $\psi' := \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi']$ with $\varphi' := \varphi_1 \wedge \varphi^{\text{relativized}}$. In this definition, $\varphi^{\text{relativized}}$ arises from φ by relativizing each quantifier of the formula to only take elements from the given structure A into account, and not elements from the added decomposition D. Consequently, ψ' defines $\mathcal{P}_1 \cap \mathcal{P}$ since $\varphi^{\text{relativized}}$ only depends on A. ◀

DMSO-logic has an alternative definition similar to a theorem of Chomsky and Schützenberger (see, for example, Harrison's book [13] for background and a proof of it) about representing a context-free language as the strings generated by (1) taking words with just brackets that are well-formed, (2) take the words of this kind that satisfy a regular property, and (3) the image of these words under a homomorphism. In the below proposition, the words of brackets are replaced by structures expanded with tree decompositions, the particular type of a bracket in combination with the regular property is replace by an MSO-property, and the homomorphism is replaced by projecting from (A, D) to A. It can be seen as moving the interpretation of the existential decomposition quantifier into the structures defined by an MSO-formula via using a larger vocabulary.

▶ **Proposition 13.** *Let $\varphi^\star$ be an MSO-formula over $\tau^\star$ for some τ that defines a property $\mathcal{P}^\star$ of structures expanded by normal tree decompositions of a bounded width $w \in \mathbb{N}$. Then the property $\mathcal{P}$ that contains all τ-structures that arise from reducing structures in $\mathcal{P}^\star$ to the relations in τ is DMSO-definable.*

5.2 Influence of the Width Parameter on Definable Properties

We denote by SO the class of all properties that are SO-definable and by SO *on* $\mathcal{C}$ the class of properties that are SO-definable on a set $\mathcal{C}$ of structures. In the same way, we define classes of properties based on MSO-definability and DMSO-definability. For structures of bounded tree width, DMSO-logic relates to the other logics discussed as follows.

▶ **Theorem 14.** *Consider a width bound $w \geq 1$ and let $\mathcal{C}$ be the set of all graphs (structures over τ_{graphs}) with tree width at most w. Then* MSO $\subsetneq$ DMSO $\subsetneq$ SO *on* $\mathcal{C}$.

Proof. MSO is separated from DMSO on strings via Fact 5 and Theorem 6; that means, by proving that languages like $a^n b^n$ are context-free, but not regular. DMSO is separated from SO on strings via languages that are not context-free, but SO-definable like $a^n b^n c^n$. The latter follows from Lemma 10 and the observation that the used tree decomposition can be

represented by a relation, which can be guessed by a second-order existential quantifier. The theorem follows from the insight that a string over characters from a fixed alphabet can be represented by a path graph without unary relations for the characters, but with spikes of varying length leaving the path's nodes to encode the characters. ◀

In the following, we take a closer look at the expressive power of DMSO-formulas and how it varies for different kinds of formulas. Remember that a DMSO-formula has the form $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$. Since the expressive power of the subformula φ, which is evaluated for a structure of bounded tree width, can be understood from the large literature on MSO's expressive power (see [15] and [5] for an overview), we concentrate on understanding the influence of the width bound w on the expressive power of formulas. First of all, since the width of the tree decomposition can also be reduced using the subformula φ, a higher built-in width bound w never results in a weaker expressive power. On the other side, in the case of strings, the equivalence to the context-free languages holds for any width bound that is at least 3. That means, width bounds beyond 3 do not result in a higher expressivity on strings. In the following, we show that this behavior does not translate to more general structures. That means, in contrast to the situation on strings, some properties of structures with a constant tree width only become DMSO-definable by increasing the width bound of the used decompositions—we cannot bound their width in terms of the width of the input structures. Interestingly, subdivided star graphs are all we need to exhibit this behavior. They have only tree width 1 and path width 2, which means that they admit width-2 tree decompositions whose underlying tree is a path.

▶ **Theorem 15.** *There is a set of structures $\mathcal{C}$ with tree width 1 and for every $w \geq 3$ a* DMSO*-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$, such that there is no* DMSO*-formula $\psi' = \exists D\,[\varphi_{\text{tw-}w'\text{-dec}} \wedge \varphi']$ with $w' \leq w - 2$ where ψ and ψ' are equivalent on C.*

The proof of Theorem 15 uses types. The *(quantifier) rank* of an MSO-formula φ, denoted by $\operatorname{qr}(\varphi)$, is the maximum number of nested quantifiers in φ. The *rank-q type* of a structure A over some vocabulary τ and a tuple $\bar{a} \in U(A)^k$ for some $k \in \mathbb{N}$ is the set of MSO-formulas $\varphi(x_1, \ldots, x_k)$ over τ of rank at most q with $A \models \varphi(\bar{a})$; we denote it by $\operatorname{tp}_q(A, \bar{a})$ and also write $\operatorname{tp}_q(A)$ for $k = 0$. The class of rank-q types for a vocabulary τ and $k \in \mathbb{N}$ is

$$\operatorname{TP}_q(\tau, k) := \{\operatorname{tp}_q(A, \bar{a}) \mid A \text{ is a } \tau\text{-structure and } \bar{a} \in U(A)^k\}\,.$$

A type only contains a finite number (depending on τ, q, and k) of non-equivalent formulas and, thus, it can be represented by a finite set of formulas; one formula for each equivalence class of formulas (see [15] for more details). Consequently, we view types as constant-size objects and algorithmic operations on them run in constant time.

We apply the below composition theorem (Fact 16) showing how the MSO-type of a structure arises from the MSO-types of substructures that are combined by identifying elements (see the survey [12] for more details on this fact). The upcoming proof does not use the algorithmic part of the fact's statement, but we use it in Section 6.

▶ **Fact 16.** *Consider a vocabulary τ, arities $k, \ell, m \in \mathbb{N}$, and a rank bound $q \in \mathbb{N}$. There is an algorithm that outputs $\operatorname{tp}_q(A \cup B, \bar{w})$ on input of $\operatorname{tp}_q(A, \bar{u}\bar{w})$ and $\operatorname{tp}_q(B, \bar{v}\bar{w})$ where A and B are τ-structures with tuples $\bar{u} = (u_1, \ldots, u_k) \in A^k$, $\bar{v} = (v_1, \ldots, v_\ell) \in B^\ell$, and $\bar{w} = (w_1, \ldots, w_m) \in (A \cap B)^m$ with $A \cap B = \{w_1, \ldots, w_m\}$.*

Proof of Theorem 15. The class $\mathcal{C}$ contains all subdivisions of stars. A star is a tree where all leaves are adjacent to the root and a subdivision of it arises by replacing edges with paths;

we call them *stars* for sake of simplicity. They have tree width 1 and, moreover, also path width 2. The paths that leave the root are called *rays*. Out of them, it will be enough to look at stars where the number of rays is bounded by the width parameter w from the theorem's statement. Formally, we use the following properties of graphs: STARS := {graph G | G is a subdivided star graph}, w-STARS := {graph G | G is a subdivided star with w rays}. The separating query singles out stars with rays of equal length: STARS-EQU := {graph G | $G \in$ STARS and all rays have the same length} and w-STARS-EQU := w-STARS $\cap$ STARS-EQU

For $w \in \mathbb{N}$ with $w \geq 3$, we claim that there is a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ defining w-STARS-EQU, but no DMSO-formula $\psi' = \exists D\,[\varphi_{\text{tw-}w'\text{-dec}} \wedge \varphi']$ with $w' \leq w-2$ defining w-STARS-EQU. This claim proves the theorem.

For the first part of the claim, φ consists of several parts combined by a conjunction; $\varphi := \varphi_{\text{STARS}} \wedge \varphi_{w\text{-LEAVES}} \wedge \varphi_{\text{EQU}}$. The MSO-formula φ_{STARS} defines the set of all subdivided stars while $\varphi_{w\text{-LEAVES}}$ defines the set of graphs with at most w degree-1 vertices. If these properties are satisfied, we are dealing with a subdivided star that has at most w rays. Based on this, the formula ψ_{EQU} accesses a decomposition D to single out the subdivided stars that have rays of equal length. First of all, it restricts D to be a tree decomposition whose underlying tree is a path (called a *path decompositions*), such that the left-most bag contains all leaves of the subdivided star. Then edges of the rays are added in an alternating way until also all edges to the root are covered. Consuming the edges in an alternating way ensures that the rays have equal length. MSO-definability of these requirements holds since w is constant.

Let $\psi' = \exists D\,[\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi']$ be a DMSO-formula that, for sake of a contradiction, defines w-STARS-EQU. Consider a subdivided star graph G with w rays, let D be a normal tree decomposition of width at most $w-2$, and let v be the root of the star. Let B_v be the highest bag in D that contains v and let v_1 to $v_{w'}$ for $w' \leq w-2$ be the other vertices of G in B_v. The bag B_v is not a leaf since, otherwise, there are edges leaving v that are not covered by the tree decompositions. Due to the same reason, there are child bags (that means, at least one child bag) that contain v. At least two rays of the star are processed by decompositions that are rooted at these children and that are disjoint except for, possibly, the root bag. This follows from the normalization condition, which makes all components defined by the tree decomposition connected. Let B_1 and B_2 be child bags of B_v (that may be the same) and D_1 and D_2 be tree decompositions inside D of rays P_1 and P_2 of the star with roots B_1 and B_2. We assume that D_2 together with P_2 is large enough, such that the smallest structure of the same rank-qr($\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi'$) type over $\tau^{\star}_{\text{graphs}}$ covers a graph (and, hence, a path) with fewer nodes. We replace (P_2, D_2) in (G, D) by this structure. The resulting structure (G', D') still satisfies $\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi'$ by the construction and the composition theorem from Fact 16 with choosing $k, \ell, m \in \mathbb{N}$ appropriately, but not all rays of G' have the same length. Thus, ψ' does not define w-STARS-EQU, a contradiction. ◀

6 Algorithmic Aspects

The satisfiability problem [17] and evaluation problem [4, 7] for MSO-logic on tree-width-bounded structures are well-studied. We extend this work by studying the decidability of the satisfiability and related problems for DMSO-logic (Section 6.1) and the complexity of its formula-evaluation problem (Section 6.2). In order to do this, formulas and structures need to be represented as strings—to be given as inputs, processed by Turing machines, and written as outputs. A formula is encoded in a direct way by encoding its symbols individually and a structure is encoded by extending the standard adjacency matrix encoding for graphs to general structures (for details of this approach see, for example, [10]).

6.1 Decidability of the Satisfiability and Related Problems

Trakhtenbrot's Theorem (for a proof see [15]) states that it is undecidable whether a given first-order formula is satisfied by a finite structure. Thus, this also transfers to MSO-logic, which generalizes first-order logic. On the other side, part of Seese's Theorem [17] is saying that the satisfiability problem for MSO-formulas is decidable if there is a bound on the tree width of the considered structures. This can be formally stated as follows.

▶ **Fact 17.** *There is a Turing machine that, given an* MSO*-formula φ over a vocabulary τ and a width $w \in \mathbb{N}$, decides whether there is a τ-structure A with* $\mathrm{tw}(A) \leq w$ *and* $A \models \varphi$.

Since DMSO-formulas are based on combining a tree width bound with an MSO-formula, Seese's Theorem extends to them. In light of the connection to context-free languages, this can be seen as generalizing the fact that the emptiness problem for context-free languages (given a context-free grammar or a pushdown automaton) is decidable.

▶ **Theorem 18.** *There is a Turing machine that, given a* DMSO*-formula ψ over some vocabulary τ, decides whether there is a τ-structure A with $A \models \psi$.*

Proof. Let $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$. As stated by Lemma 2, a structure A that is expanded by a tree decomposition of width at most w has tree width at most $w + 2$. Thus, in order to decide whether ψ is satisfied by a τ-structure, it is enough to decide whether the inner part of the formula, which is $\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi$, is satisfied by a $\tau^\star$-structure of tree width at most $w + 2$. This is possible due to Fact 17. ◀

Since MSO-logic is closed under taking Boolean connectives, the decision procedure that we get from Seese's Theorem extends to the question whether any Boolean combination of two given MSO-formulas is satisfied by a structure of bounded tree width. Due to its equivalence on strings to the context-free languages, these closure properties do not transfer to DMSO-logic. DMSO-logic inherits the following undecidability results from the context-free languages (for proofs of them see, for example, [13]).

▶ **Proposition 19.** *Each of the following questions is not decidable for given* DMSO*-formulas ψ_1 and ψ_2 defining properties $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively: $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$? $|\mathcal{P}_1 \cap \mathcal{P}_2| = \infty$? $\mathcal{P}_1 \cap \mathcal{P}_2 \in$ DMSO? $\mathcal{P}_1 \subseteq \mathcal{P}_2$? $\mathcal{P}_1 = \mathcal{P}_2$?*

6.2 Computational complexity of the Evaluation Problems

The task of evaluating an MSO-formula for a structure A comes in many flavors; leading to different insights into the complexity of evaluating MSO-formulas. If both the formula and the structure are part of the input, then the problem is complete for polynomial space [20]. If the formula is fixed (that means, we look at a problem based on an MSO-definable property), then we still have problems complete for any level of the polynomial hierarchy (see [15] for a proof of this). The complexity of the problem drops significantly if, in addition, the tree width of input structures is bounded by some constant: deciding $A \models \varphi$ can be done in polynomial time for input structures whose tree width is bounded by some constant and even algorithms running in linear-time [4] or having a logarithmic memory footprint exist [7]. During the course of the present section, we show that deciding $A \models \psi$ can also be done in polynomial time for every fixed DMSO-formula ψ (where we do not need to make the tree width bound explicit since it is implicitly given as part of the formula).

▶ **Theorem 20.** *For every* DMSO*-formula ψ, there is a polynomial-time algorithm that, given a structure A, decides $A \models \psi$.*

Proof. We consider a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ over some vocabulary τ and set $q := \mathrm{qr}(\varphi)$. While $\exists D$, the decomposition quantifier, ranges over all tree decompositions of a given structure A, the MSO-formula $\varphi_{\text{tw-}w\text{-dec}}$ only evaluates to true for structures expanded by a tree decompositions that is normal and has width w. Thus, in order to test whether $A \models \psi$ holds for a given τ-structure, it is enough to test whether $(A, D) \models \varphi$ holds for some width-w normal tree decomposition D of A. Instead of directly working with MSO-formulas, we work with types as already used in Section 5. That means, instead of testing whether $(A, D) \models \varphi$ holds for a width-w normal tree decompositions, we compute

$$\mathrm{TP}^\star_q(A) := \{\mathrm{tp}_q(A^\star) \mid A^\star = (A, D) \text{ is a } \tau^\star\text{-structure with } (A, D) \models \varphi_{\text{tw-}w\text{-dec}}\},$$

which has constant size by definition. Then deciding $A \models \psi$ is equivalent to testing whether φ is equivalent to a formula from θ for some $\theta \in \mathrm{TP}^\star_q(A)$. Since this is a constant-time operation, we direct our attention to computing $\mathrm{TP}^\star_q(A)$.

We compute $\mathrm{TP}^\star_q(A)$ via an inductive approach along growing substructures. It computes types for substructures of A by applying the type-composition theorem from Fact 16 to the types that are already computed for smaller substructures. Since the types are based on both the structure and a tree decomposition for it, while doing this, we enrich the substructures of A by parts of candidate tree decompositions that expand them. Overall, we take the MSO-types of all width-w normal tree decompositions for A into account, but without constructing decompositions explicitly; this amounts to a dynamic-programming approach along appropriate substructures.

We use some terminology adapted from [7] to define the substructures and the types of substructures we consider: Set $G := G(A)$ and $U := U(A)$. A *descriptor* in A is a tuple (σ_i, v_i) consisting of a set $\sigma_i \subseteq U$ with $|\sigma_i| \leq w + 1$, called the *separator*, and an element $v_i \in U \setminus \sigma_i$, called the *(component) selector*. We denote by α_i the vertex set of the component of $G[U \setminus \sigma_i]$ that contains v_i and set $\gamma_i := \sigma_i \cup \alpha_i$. For each (σ_i, v_i) with $A_i := A[\gamma_i]$, we consider some (ordered) sequence $\bar{\sigma}_i$ of the elements in σ_i and compute

$$\begin{aligned}\mathrm{TP}^\star_q(A_i, \sigma_i) := \{\mathrm{tp}_q(A^\star_i, \bar{\sigma}^\star_i) \mid A^\star_i = (A_i, D_i) \text{ is a } \tau^\star\text{-structure with} \\ (A_i, D_i) \models \varphi_{\text{tw-}w\text{-dec}}, \sigma_i \subseteq \beta(\mathrm{r}(D_i)) \text{ and } \bar{\sigma}^\star_i = (\bar{\sigma}_i, \mathrm{r}(D_i))\},\end{aligned}$$

which is only empty if there is no width-w normal tree decomposition of the described kind. Since $G[A]$ is connected (otherwise, A does not have a normal tree decomposition by definition), we have $\mathrm{TP}^\star_q(A) = \mathrm{TP}^\star_q(A_{i_0}, \bar{\sigma}_{i_0})$ for a descriptor (σ_{i_0}, v_{i_0}) with $\sigma_{i_0} = \emptyset$ and an arbitrary, but fixed, element $v_{i_0} \in U$. Thus, in particular, computing all $\mathrm{TP}^\star_q(A_i, \bar{\sigma}_i)$ for descriptors (σ_i, v_i) proves the theorem.

Let $(\sigma_1, v_1), \ldots, (\sigma_n, v_n)$ be the sequence of all descriptors sorted by $|\gamma_i|$, the size of their cones, with higher priority and by $|\alpha_i|$, the size of their components, with lower priority. This sequence is computable in polynomial time since w is constant. We compute $\mathrm{TP}^\star_q(A_i, \bar{\sigma}_i)$ for each (σ_i, v_i) in the order of this sequence. If $|\gamma_i| \leq w + 1$, we can compute it in constant time; there are only a constant number of ways to turn the structure A_i into a structure that is expanded by a width-w normal tree decomposition. If $|\gamma_i| > w + 1$, we consider each $\beta_{i,j} \subseteq \gamma_i$ with $\sigma_i \subseteq \beta_{i,j}$ and $|\beta_{i,j}| \leq w + 1$, which are the candidate root bags of a tree decomposition for A_i, and compute

$$\begin{aligned}\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j}) := \{\mathrm{tp}_q(A^\star_i, \bar{\sigma}^\star_i) \mid A^\star_i = (A_i, D_i) \text{ is a } \tau^\star\text{-structure with} \\ (A_i, D_i) \models \varphi_{\text{tw-}w\text{-dec}}, \beta_{i,j} = \beta(\mathrm{r}(D_i)) \text{ and } \bar{\sigma}^\star_i = (\bar{\sigma}_i, \mathrm{r}(D_i))\}.\end{aligned}$$

Then $\mathrm{TP}^\star_q(A_i, \sigma_i)$ can be computed in polynomial time from all $\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j})$ by cycling through the candidate root bags $\beta_{i,j}$ of size at most $w + 1$.

For each $\beta_{i,j}$, we consider the components $\alpha'_1, \ldots, \alpha'_m$ of $A_i \setminus \beta_{i,j}$ and define $A'_m = A_i$ as well as $A'_k = A'_{k+1} \setminus \alpha'_{k+1}$ for all $k \in \{1, \ldots, m-1\}$. With this definition, we have $\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j}) = \mathrm{TP}^\star_q(A'_m, \sigma_i, \beta_{i,j})$. In order to compute this class, we compute the $\mathrm{TP}^\star_q(A'_k, \sigma_i, \beta_{i,j})$ along growing k. For each k with component α'_k, we consider all combinations of separators $\sigma'_{i,j,k} \subseteq \beta_{i,j}$ and selector vertices $v'_{i,j,k} \in \alpha'_j$ that describe this component. Then we use the type-composition theorem to compute the types of A'_k that we get from normal width-w tree decompositions where the intersection of the root bag $\beta_{i,j}$ and the bag below it with component α'_k is $\sigma'_{i,j,k}$. Fact 16 can be applied since the overlap of A'_{k-1} and the newly added part has constant size and this still holds if we take the two additional nodes from the tree decomposition into account. Since the type-composition is a constant-time operation, the algorithm runs in polynomial time. ◀

7 Conclusion

The present paper proposed DMSO-logic for capturing the notion of context-free graph properties. To support this, we proved results about its language-theoretic, model-theoretic, and algorithmic aspects. Notably, it corresponds to the context-free languages on strings and has a formula-evaluation procedure that runs in polynomial time for every fixed formula. Moreover, we observed that the built-in width bound crucially influences expressive power.

There are a number of possible directions for future work: First of all, it would be interesting to know how DMSO-logic relates to the different notions of context-free properties of trees [9] and, more generally, graphs [5]. Second, it would be interesting to obtain more efficient formula-evaluation procedures. Concrete open questions are whether deciding $A \models \psi$ is fixed-parameter tractable when taking ψ as the parameter and whether it is in LOGCFL (the class of problems logspace-reducible to the context-free languages) for every fixed formula. Both are reasonable conjectures in the light of previous work on the formula-evaluation problem for MSO-logic [4, 7].

References

1. Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proceedings of the 19th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 641–650. SIAM, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347153`.
2. Mikołaj Bojańczyk and Michał Pilipczuk. Definability equals recognizability for graphs of bounded tree width. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016)*. IEEE Computer Society, 2016. to appear.
3. J. Richard Büchi. Weak second-order arithmetic and finite automata. *Math. Logic Quart.*, 6(1–6):66–92, 1960.
4. Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B*, pages 193–242. Elsevier and MIT Press, 1990.
5. Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*. Cambridge University Press, 2012. URL: `http://hal.archives-ouvertes.fr/hal-00646514/fr/`.
6. John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970. `doi:10.1016/S0022-0000(70)80041-1`.
7. Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152, 2010. `doi:10.1109/FOCS.2010.21`.

8 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961. doi:10.2307/1993511.

9 Joost Engelfriet and Erik Meineche Schmidt. IO and OI. I. *Journal of Computer and System Science*, 15(3):328–353, 1977. doi:10.1016/S0022-0000(77)80034-2.

10 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

11 Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *Journal of the ACM*, 59(5):27:1–27:64, 2012. doi:10.1145/2371656.2371662.

12 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 181–206. American Mathematical Society, 2011. URL: http://www.automata.rwth-aachen.de/~grohe/pub/grokre11.pdf.

13 Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

14 Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In *Proceedings of CSL 1994*, volume 933, pages 205–216. Springer, 1995.

15 Leonid Libkin. *Elements Of Finite Model Theory*. Springer, 2004. doi:10.1002/malq.19600060105.

16 J. Mezei and J.B. Wright. Algebraic automata and context-free sets. *Inform. and Control*, 11(1–2):3–29, 1967. doi:10.1016/S0019-9958(67)90353-1.

17 D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic*, 53(2):169–195, 1991. doi:10.1016/0168-0072(91)90054-P.

18 J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968. doi:10.1007/BF01691346.

19 Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.

20 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the 14th annual ACM symposium on Theory of computing (STOC 1982)*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.