New Algorithm for Weak Monadic Second-Order Logic on Inductive Structures

Tobias Ganzow and Łukasz Kaiser

Mathematische Grundlagen der Informatik, RWTH Aachen University, Germany
{ganzow,kaiser}@logic.rwth-aachen.de

Abstract. We present a new algorithm for model-checking weak monadic second-order logic on inductive structures, a class of structures of bounded clique width. Our algorithm directly manipulates formulas and checks them on the structure of interest, thus avoiding both the use of automata and the need to interpret the structure in the binary tree. In addition to the algorithm, we give a new proof of decidability of weak MSO on inductive structures which follows Shelah's composition method. Generalizing this proof technique, we obtain decidability of weak MSO extended with the unbounding quantifier on the binary tree, which was open before.

1 Introduction

Monadic second-order logic (MSO) is an extension of first-order logic in which quantification over subsets of the universe is allowed. Using the connection to automata, it was shown by Büchi that MSO is decidable on $(\omega, <)$ [1], and by Rabin [2] that it is decidable on the infinite binary tree. Using interpretations, this result has been extended from the binary tree to all structures of bounded clique-width [3], showing that MSO is decidable on a large class of structures.

In practical applications such as verification of software systems or hardware, the domain of interest is often finite but not a priori bounded in size, and thus many verification problems can be naturally formalized in *weak* MSO, a fragment of MSO which allows only quantification over *finite* subsets of the universe. The best known tool for model-checking WMSO, Mona, has been used to verify hardware [4] and pointer manipulating programs [5], and is part of software verification systems. e.g. [6]. To use Mona for program verification it is necessary to interpret the structure of interest in the binary tree, which is often a cause of inefficiency. Moreover, since Mona is based on automata, it is challenging to use it for verifying properties which mix terms from different theories.

These problems motivated us to devise an algorithm for weak MSO model-checking, together with a proof of its correctness, that exploits logical tools and structural aspects of the models rather than being based on automata. Our algorithm works on the general class of inductive structures which comprise classical structures such as $(\omega, <)$ and the binary tree as well as practically relevant structures such as doubly-linked lists or lists of lists. Inductive structures can in fact be encoded in the binary tree, but we avoid this both because it is a source of inefficiency and because our algorithm can be easily formulated directly for arbitrary inductive structures. Moreover, our algorithm is not based on automata and in each step only manipulates a set of formulas. This makes it well-suited to be a part of a larger verification system or SMT solver, since it is amenable to Nelson-Oppen style combination with other theories.

In addition to the algorithm, we present a new proof of decidability of weak MSO on inductive structures. Our proof follows the *composition method*, which was used by Shelah [7] (see also [8]) to show decidability of unrestricted MSO on $(\omega, <)$ and other countable linear orders, as well as by Läuchli [9] in his proof of the decidability of the weak MSO theory of linear orders. Both proofs are based on the enumeration of all types of a certain quantifier rank, and can therefore not be used as a basis for an algorithm. As yet, the composition method has not been generalized to unrestricted MSO on the binary tree, and the question about a composition-based proof of decidability of MSO on the binary tree is, in fact, considered a major open problem, because of its close relationship to the challenge of understanding the algebraic structure of regular tree languages.

A thorough overview of applications of the composition method for obtaining decidability results for MSO on various classes of structures is given in [10]. Furthermore, see [11] for an account on the evolution of the field.

We exploit that, in contrast to $(\omega, <)$, the weak MSO theory of the binary tree is simpler than its unrestricted MSO theory, and show, using decompositions of weak MSO formulas, that the model-checking problem for weak MSO on inductively defined structures can be reduced to determining the winner of a finite reachability game. However, the worst-case complexity of checking weak MSO sentences on $(\omega, <)$ is already non-elementary. Therefore the decompositions of the checked formula, and hence the game graph, can be huge. As the bound is tight, this cannot be avoided in general, but a preliminary implementation of the algorithm shows that the approach works on basic examples.

We also show that more general quantifiers can be integrated in our approach by proving that WMSO with the unbounding quantifier, a logic which has recently been shown to be decidable on labelings of $(\omega, <)$ [12], is also decidable on inductive structures, in particular on the binary tree.

2 Preliminaries

A relational structure $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_K^{\mathfrak{A}})$ over the signature $\tau = \{R_1, \ldots, R_K\}$ (where each R_i has an associated arity r_i) consists of the universe A and relations $R_i^{\mathfrak{A}} \subseteq A^{r_i}$. We say that $\mathfrak{A} \subseteq \mathfrak{B}$ if the universe $A \subseteq B$ and each $R_i^{\mathfrak{A}} \subseteq R_i^{\mathfrak{B}}$. For a subset $B \subseteq A$ we define $\mathfrak{A} \cap B$ as the structure with universe B and relations $R_i^{\mathfrak{A}} \cap B^{r_i}$. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic if there exists a bijection $\pi : A \to B$ between their universes such that $(a_1, \ldots, a_{r_i}) \in R_i^{\mathfrak{A}} \iff (\pi(a_1), \ldots, \pi(a_{r_i})) \in R_i^{\mathfrak{B}}$. We write $[k]$ for the set $\{1, \ldots, k\}$, and, for a given set A, we write A^* for the set of all finite sequences of elements of A.

Weak monadic second-order logic (WMSO) extends first-order logic by quantification over *finite subsets* of the universe. In WMSO, first-order variables

$x, y, \dots$ are interpreted as elements, and set variables $X, Y, \dots$ as *finite* subsets of the universe. Set variables are capitalized to distinguish them from first-order variables. The atomic formulas are $R_i(\overline{x})$, $x = y$, $x \in X$, and $\bot$ for *false* and $\top$ for *true*. All other formulas are built from atomic ones by applying Boolean connectives and universal and existential quantifiers for both kinds of variables.

2.1 Inductive Structures

We investigate weak monadic second-order logic on *inductive structures*. One way to characterize such structures is using the notion of bounded clique-width decomposition: inductive structures admit a bounded clique-width decomposition with regular labels. However, to remain self-contained and due to the way our algorithms work, we give another definition using a system of equations, similar to the definition of vertex replacement (VR) graphs but strictly monotone.

In the following, we will frequently speak of *indexed structures* and *indexed elements*. The latter are elements paired with a finite word (called index) over a specific alphabet Σ. An indexed structure consists of a universe of indexed elements. We usually identify a plain structure with the indexed structure in which all elements are indexed by ε (i.e. the empty word).

Definition 1. *Given, for each $R_i \in \tau$, a function $f_i : \{1, \dots, k\}^{r_i} \to \{\bot, \top\}$, and indexed structures $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ over $\Sigma \supseteq [k]$, we define the* (k-ary) disjoint sum with connections $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \dots, R_k^{\mathfrak{B}})$*, denoted $\bigoplus_{\overline{f}}(\mathfrak{A}_1, \dots, \mathfrak{A}_k)$, by:*

- *$B := \{(a, jw) : (a, w) \in A_j,\ j \in [k]\}$ and*
- *$\big((b_1, j_1 w_1), \dots, (b_{r_i}, j_{r_i} w_{r_i})\big) \in R_i^{\mathfrak{B}}$ if*
  - *$|\{j_1, \dots, j_{r_i}\}| = 1$ and $(b_1, \dots, b_{r_i}) \in R_i^{\mathfrak{A}_{j_1}}$ or*
  - *$|\{j_1, \dots, j_{r_i}\}| > 1$ and $f_i(j_1, \dots, j_{r_i}) = \top$.*

That is, $\mathfrak{B}$ is constructed by taking the disjoint union of the structures $\mathfrak{A}_j$, and adding tuples spanning multiple components according to the given functions f_i. It is implicit in the definition that unary relations are only inherited from the components, whereas only at least binary relations are augmented with additional tuples. Intuitively, the indices keep track of the origin of elements. We let $\mathfrak{B}[j] := \mathfrak{B} \cap \{(b, w) \in B : w = jw'\}$ denote the j-th *component* of the disjoint sum. Note that, as expected, $\mathfrak{B}[j]$ is isomorphic to $\mathfrak{A}_j$ via $\pi_j : (b, jw') \mapsto (b, w')$. Furthermore, defining $\mathfrak{B}[\varepsilon] := \mathfrak{B}$, the notation naturally extends to $\mathfrak{B}[wj] := (\mathfrak{B}[w])[j] = \mathfrak{B} \cap \{(b, v) \in B : v = wjw'\}$.

Example 2. Given $f(1, 2) = \top$ and $f(i, j) = \bot$ otherwise, $\begin{tikzcd} \bullet \\ \bullet \arrow[u] \end{tikzcd} \oplus_f \begin{tikzcd} \bullet \arrow[d] \\ \bullet \end{tikzcd} = \begin{tikzcd} 1\,\bullet \arrow[r] \arrow[rd] & \bullet\,2 \arrow[d] \\ 1\,\bullet \arrow[u] \arrow[r] \arrow[ru] & \bullet\,2 \end{tikzcd}$.

Definition 3. *A system of* structure equations $\mathcal{D}$ *over τ has the form*

$$\mathcal{D} = \begin{cases} \Lambda^1 = \mathfrak{A}_1^1 \oplus \mathfrak{A}_2^1 \oplus \dots \oplus \mathfrak{A}_{k_1}^1 & \textit{with } f_1^1, \dots, f_K^1 \\ \vdots \qquad\qquad \vdots & \qquad \vdots \\ \Lambda^n = \mathfrak{A}_1^n \oplus \mathfrak{A}_2^n \oplus \dots \oplus \mathfrak{A}_{k_n}^n & \textit{with } f_1^n, \dots, f_K^n \end{cases}$$

where each $\mathfrak{A}^i_j$ is either a finite *structure or one of the formal variables $\Lambda^1, \ldots, \Lambda^n$ and each f^i_j is a function $\{1, \ldots, k_i\}^{r_j} \to \{\bot, \top\}$. We write $\lambda(i,j) = m$ if $\mathfrak{A}^i_j = \Lambda^m$ and $\lambda(i,j) = Fin$ otherwise. Let $\overline{\mathfrak{B}} = (\mathfrak{B}_1, \ldots, \mathfrak{B}_n)$ be relational structures to substitute for variables on the right-hand side of $\mathcal{D}$. Then, we define the new left-hand side structures $(\mathfrak{C}_1, \ldots, \mathfrak{C}_n) = \mathcal{D}(\overline{\mathfrak{B}})$ by:*

$$\mathfrak{C}_i = \bigoplus\nolimits_{\overline{f}^i}(\mathfrak{D}_1, \ldots, \mathfrak{D}_{k_i}) \text{ where } \mathfrak{D}_j = \begin{cases} \mathfrak{A}^i_j & \text{if } \lambda(i,j) = Fin, \\ \mathfrak{B}_k & \text{if } \lambda(i,j) = k\,. \end{cases}$$

We say that a tuple $\overline{\mathfrak{B}}$ of structures *satisfies* $\mathcal{D}$ if $\mathcal{D}(\overline{\mathfrak{B}}) = \overline{\mathfrak{B}}$. Observe that the operator $\overline{\mathfrak{B}} \mapsto \mathcal{D}(\overline{\mathfrak{B}})$, mapping n-tuples of structures to new n-tuples of structures as defined above, is monotone since it only adds elements to the universe and tuples to relations. Hence, it has a unique least fixed-point $(\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$, i.e. a minimal tuple of structures that satisfies $\mathcal{D}$ and which we refer to by $\mathcal{S}(\mathcal{D})$. We denote the i-th structure of the fixed-point by $\mathcal{S}_i(\mathcal{D})$, and we call a structure $\mathfrak{A}$ *inductive* if and only if there exists a system of equations $\mathcal{D}$ such that $\mathfrak{A}$ is isomorphic to some $\mathcal{S}_i(\mathcal{D})$.

Let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$. By definition, each $\mathfrak{A}_m$ is an indexed structure over $\Sigma = [\max(k_1, \ldots, k_n)]$ obtained as a (k_m-ary) disjoint sum $\bigoplus_{\overline{f}^m}(\mathfrak{D}_j)_{j \in [k_m]}$ with additional tuples spanning components according to $\mathcal{D}$, and hence, for each $j = 1, \ldots, k_m$, the component $\mathfrak{A}_m[j]$ is either isomorphic to the finite structure $\mathfrak{A}^m_j$ given in $\mathcal{D}$ if $\lambda(m,j) = \mathrm{Fin}$ or to $\mathfrak{A}_{\lambda(m,j)}$ otherwise. For easier referencing, we will partition the sets of indices into $\mathrm{Fin}_i = \{j : \lambda(i,j) = \mathrm{Fin}\}$, and $\Delta_i = \{j : \lambda(i,j) \neq \mathrm{Fin}\}$. Furthermore, for an indexed element $(a,w) \in A_i$, the *depth* of $(a,w) \in A_i$ is defined as $\mathrm{dp}_i(a,w) = |w|$, and the depth of a set is the maximal depth of its elements, $\mathrm{dp}_i(S) = \max\{\mathrm{dp}_i(s) : s \in S\}$.

Example 4. The system defining the infinite binary tree $\mathfrak{T}_2$ with prefix ordering and unary predicates S_0 and S_1 for the left and right successor is:

$$\begin{aligned} \Lambda^1 &= \big(\{\bullet\}, S_0 = \emptyset, S_1 = \emptyset, < = \emptyset\big) \quad \oplus \Lambda^2 \oplus \Lambda^3 \text{ with } f_< \\ \Lambda^2 &= \big(\{\bullet\}, S_0 = \{\bullet\}, S_1 = \emptyset, < = \emptyset\big) \oplus \Lambda^2 \oplus \Lambda^3 \text{ with } f_< \\ \Lambda^3 &= \big(\{\bullet\}, S_0 = \emptyset, S_1 = \{\bullet\}, < = \emptyset\big) \oplus \Lambda^2 \oplus \Lambda^3 \text{ with } f_< \end{aligned}$$

where $f_<(i,j) = \top$ if $i = 1$ and $j \in \{2,3\}$ and $\bot$ in all other cases. Note that, by definition, the functions must be given only for tuples where at least two arguments differ. Therefore we give no functions for S_0 and S_1—predicates are determined solely by the right-hand side structures, as depicted in Figure 1.

As another example, we give a system defining a list of lists with two order relations, S on the primary list and L on the other lists, as depicted in Figure 2.

$$\begin{aligned} \Lambda^1 &= \big(\{\bullet\}, R_L = \emptyset, R_S = \emptyset\big) \oplus \Lambda^1 \oplus \Lambda^2 \text{ with } f^1_L, f^1_S \\ \Lambda^2 &= \big(\{\bullet\}, R_L = \emptyset, R_S = \emptyset\big) \oplus \Lambda^2 \qquad\quad \text{with } f^2_L, f^2_S \end{aligned}$$

where $f^1_L(1,2) = f^1_S(1,3) = \top$ and $f^2_L(1,2) = \top$, and $f^k_r(i,j) = \bot$ in other cases.

Observe that in both examples above a direct successor relation is definable in WMSO from the constructed orderings.

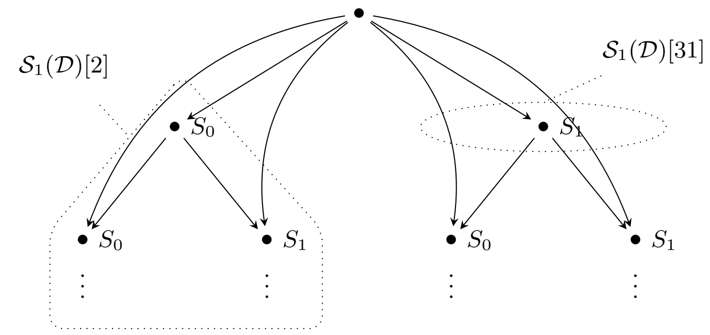
$\mathcal{S}_1(\mathcal{D})[2]$
$\mathcal{S}_1(\mathcal{D})[31]$
S_0
S_1
S_0
S_1
S_0
S_1

Fig. 1. Inductive definition of the binary tree $\mathfrak{T}_2 \cong \mathcal{S}_1(\mathcal{D})$

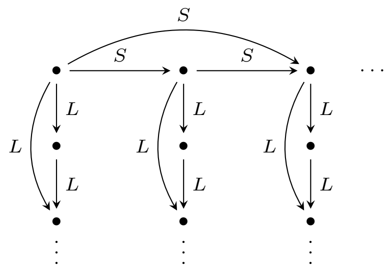

Fig. 2. Inductive definition of the infinite list of lists

2.2 Formulas with Restricted Variables

Intuitively, inductive structures are disjoint sums of other inductive structures with added relation tuples, and thus naturally decompose into *components*. When writing formulas over such structures, it is often convenient to restrict specific variables to specific components of the universe. Here we introduce related notions and a procedure to split variables so as to convert a formula into one that only contains variables restricted to disjoint parts of the universe.

Formulas with restricted variables of k kinds are defined in the same way as WMSO formulas, but in addition to the standard first- and second-order variables $x_1, x_2, \ldots$ and $X_1, X_2, \ldots$ we allow to write restricted variables $x_1^i, x_2^i, \ldots$ and $X_1^i, X_2^i, \ldots$ for $i = 1, \ldots, k$. (We use superscripts to distinguish restricted variables.) Given a structure $\mathfrak{A}$, a partition of the universe $A = A^1 \cup \cdots \cup A^k$ into k pairwise disjoint sets $A^1, \ldots, A^k$ gives rise to the so-called *partitioned structure* $\mathfrak{A}_{\langle A^1,\ldots,A^k\rangle}$. We interpret formulas with restricted variables on such partitioned structures, and intuitively x^i and X^i are understood as referring only to the i-th component A^i. More formally, we define the semantics of formulas with restricted variables on structures with partitioned universe in the standard way, with the additional rule that $\mathfrak{A}_{\langle A^1,\ldots,A^k\rangle} \models \exists X^i \varphi(X^i)$ if and only if there exists a $U \subseteq A^i$ (instead of a $U \subseteq A$) for which $\mathfrak{A}_{\langle A^1,\ldots,A^k\rangle} \models \varphi(U)$. The definition for $\forall X^i$ and first-order quantification is analogous. The interpretation of free restricted variables follows the same intuition, however, for the sake of clarity, we only allow free second-order variables.

Quantifier rank of formulas plays an important role in our proofs, and we extend this notion to formulas with restricted variables. Classically, the quantifier rank of a formula φ, $\mathrm{qr}(\varphi)$, is defined to be 0 if φ is an atomic formula, the maximum of the quantifier ranks of the conjuncts if φ is a Boolean combination and the rank of the quantified formula plus 1 if φ starts with a quantifier. We extend this notion to a formula φ with restricted variables so that $\mathrm{qr}^i(\varphi)$ counts only the nesting of quantified variables restricted to i:

- $\mathrm{qr}^i(\varphi) = 0$ if φ is an atomic formula,
- $\mathrm{qr}^i(\neg\varphi) = \mathrm{qr}^i(\varphi)$,
- $\mathrm{qr}^i(\varphi) = \max(\mathrm{qr}^i(\psi), \mathrm{qr}^i(\vartheta))$ if $\varphi = \psi \wedge \vartheta$ or $\varphi = \psi \wedge \vartheta$,
- $\mathrm{qr}^i(\exists X^j \varphi) = \mathrm{qr}^i(\exists x^j \varphi) = \mathrm{qr}^i(\forall X^j \varphi) = \mathrm{qr}^i(\forall x^j \varphi) = \begin{cases} \mathrm{qr}^i(\varphi) + 1 & \text{if } j = i \\ \mathrm{qr}^i(\varphi) & \text{otherwise.} \end{cases}$

Finally, the *restricted quantifier rank* $\mathrm{qr}^*(\varphi)$ is defined as the maximum over quantifier ranks restricted to the components: $\mathrm{qr}^*(\psi) = \max\{\mathrm{qr}^i(\psi) : 1 \leq i \leq k\}$.

2.3 Splitting Variables

Each formula of monadic second-order logic (with free second-order variables only) can be transformed into an equivalent formula in which all variables are restricted. The procedure $\texttt{split}_k$ below computes, for a formula φ with variables $\overline{X}, \overline{x}$ and a fixed k, a formula ψ with variables $\overline{X}^i, \overline{x}^i$, $i = 1, \ldots, k$ such that $\mathfrak{A}, \overline{V} \models \varphi$ if and only if $\mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \overline{V} \models \psi$ for any partition $A^1, \ldots, A^k$ of the universe of $\mathfrak{A}$ and any interpretation of the free second-order variables by sets $\overline{V}$; if a free variable X is assigned the set V, then the corresponding restricted variables X^i are assigned the sets $V \cap A^i$. In the notation used in procedure $\texttt{split}_k$, we allow to substitute a sum, e.g. $X \cup Y$ for a second-order variable Z. This should be understood as replacing each atom $z \in Z$ by $z \in X \vee z \in Y$ (and $Z \leftarrow \emptyset$ means substituting $z \in Z$ by $\bot$).

By induction on the structure of the formulas and using the above definition of $\texttt{split}_k(\varphi)$, we directly obtain the following lemma.

Lemma 5. *For every weak MSO formula φ with free monadic second-order variables only, every structure $\mathfrak{A}$, every partition $(A^1, \ldots, A^k)$ of the universe of $\mathfrak{A}$, and every assignment of sets $\overline{V}$ to the free second-order variables of φ, we have $(\mathfrak{A}, \overline{V}) \models \varphi$ if and only if $(\mathfrak{A}_{\langle A^1, \ldots, A^k \rangle}, \overline{V}) \models \texttt{split}_k(\varphi)$. Moreover, $\mathrm{qr}^*(\texttt{split}_k(\varphi)) \leq \mathrm{qr}(\varphi)$.*

3 Decomposing Formulas

Given a system of equations which defines an inductive structure, we can decompose a WMSO formula into a Boolean combination of formulas to be checked on the constituent structures.

Procedure $\mathrm{split}_k(\varphi)$

```
case $\varphi$ contains a free (unrestricted) variable $X$ return $\mathtt{split}_k(\varphi[X \leftarrow \bigcup_i X^i])$;
case $\varphi$ is an atom return $\varphi$;
case $\varphi = \neg\psi$ return $\neg\mathtt{split}_k(\psi)$;
case $\varphi = \varphi_1 \vee \varphi_2$ return $\mathtt{split}_k(\varphi_1) \vee \mathtt{split}_k(\varphi_2)$;
case $\varphi = \varphi_1 \wedge \varphi_2$ return $\mathtt{split}_k(\varphi_1) \wedge \mathtt{split}_k(\varphi_2)$;
case $\varphi = \exists x\psi$ return $\bigvee_{i=1,\dots,k} \exists x^i \mathtt{split}_k(\psi)[x \leftarrow x^i]$;
case $\varphi = \forall x\psi$ return $\bigwedge_{i=1,\dots,k} \forall x^i \mathtt{split}_k(\psi)[x \leftarrow x^i]$;
case $\varphi = \exists X\psi$ return $\exists X^1 \dots X^k \mathtt{split}_k(\psi)[X \leftarrow \bigcup_i X^i]$;
case $\varphi = \forall X\psi$ return $\forall X^1 \dots X^k \mathtt{split}_k(\psi)[X \leftarrow \bigcup_i X^i]$;
```

Definition 6. *Let $\mathcal{D}$ be a system of n structure equations such that k_i structures appear on the right-hand side of the i-th equation. Let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$ and let φ be a WMSO formula with free variables $X_1, \dots, X_r$ (note that it has no free first-order variables). For each $m \in [n]$, a* $\mathcal{D}_m$-decomposition *of φ is a sequence of k-tuples ($k = k_m$) of formulas $(\psi_1^1, \dots, \psi_k^1), \dots, (\psi_1^l, \dots, \psi_k^l)$ such that the free variables of each ψ_j^i are included in $X_1, \dots, X_r$, $\mathrm{qr}(\psi_j^i) \le \mathrm{qr}(\varphi)$, and*

$$\mathfrak{A}_m, \overline{V} \models \varphi \iff \text{for some } i \in [l] \text{ and each } j \in [k] \quad \mathfrak{A}_m[j], \overline{V} \cap \mathfrak{A}_m[j] \models \psi_j^i.$$

The following theorem is the main result used to prove the correctness of our algorithm. Let us remark that it can be obtained from more general composition theorems of Shelah [7], but those theorems do not yield a practical algorithm.

Theorem 7. *For every WMSO formula φ, system of n structure equations $\mathcal{D}$, and $m \in [n]$, there exists an effectively computable $\mathcal{D}_m$-decomposition of φ.*

Note that our notion of $\mathcal{D}_m$-decompositions corresponds to reduction sequences introduced by Feferman and Vaught for FO. An example of how to compute these for MSO in a special case was described in [11]. The rest of this section is devoted to a proof of the above theorem in a more general setting which yields a basic building block for the model-checking algorithm. Towards this, we introduce a new normal form of WMSO formulas, which we call TNF, the *type normal form.* TNF is in a sense a converse of the prenex normal form since quantifiers are pushed as deep inside the formulas as possible.

3.1 Type Normal Form

For a set of formulas Φ we denote by $\mathcal{B}^+(\Phi)$ all positive Boolean combinations of formulas from Φ, i.e. formulas given by $\mathcal{B}^+(\Phi) = \Phi \mid \mathcal{B}^+(\Phi) \vee \mathcal{B}^+(\Phi) \mid \mathcal{B}^+(\Phi) \wedge \mathcal{B}^+(\Phi)$. A formula is in TNF if and only if it is a positive Boolean combination of formulas of the following form

$$\begin{aligned} \tau = R_i(\overline{x}) \mid \neg R_i(\overline{x}) \mid x = y \mid x \neq y \mid x \in X \mid x \notin X \\ \mid \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau) \end{aligned}$$

satisfying the following crucial constraint: in $\exists x \mathcal{B}^+(\tau_i)$, $\exists X \mathcal{B}^+(\tau_i)$, $\forall x \mathcal{B}^+(\tau_i)$, and $\forall X \mathcal{B}^+(\tau_i)$ the free variables of *each* τ_i appearing in the Boolean combination *must contain* x, or respectively X.

We claim that for each formula φ there exists an equivalent formula ψ in TNF such that $\mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi)$ (and $\mathrm{qr}^*(\psi) \leq \mathrm{qr}^*(\varphi)$ for formulas with restricted variables) and the set of atoms of ψ is a subset of the atoms of φ. The procedure $\mathtt{TNF}(\varphi)$ computes such a formula ψ given a formula φ in negation normal form. Note that it uses sub-procedures `DNF` and `CNF` which, given a Boolean combination of formulas, convert it to disjunctive or conjunctive normal form. As an example, consider $\varphi = \exists x\big(P(x) \land (Q(y) \lor R(x))\big)$; $\mathtt{TNF}(\varphi) = \big(Q(y) \land \exists x P(x)\big) \lor \exists x\big(P(x) \land R(x)\big)$.

Theorem 8. *The formula $\psi = \mathtt{TNF}(\varphi)$ is in TNF, equivalent to φ, its atoms and free variables are included in the ones of φ and $\mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi)$. If φ contains restricted variables, then $\mathrm{qr}^*(\psi) \leq \mathrm{qr}^*(\varphi)$.*

Proof. We proceed inductively on the structure of φ. For literals all the claims are trivial since `TNF` is an identity. For Boolean combinations of formulas, the procedure `TNF` only calls itself recursively, thus all claims of the theorem follow inductively as well.

Consider the case when $\varphi = \exists x \psi$ and $\mathtt{DNF}(\mathtt{TNF}(\psi)) = \bigvee_i (\bigwedge_j \psi^i_j)$. We convert $\mathtt{TNF}(\psi)$ to disjunctive normal form in this case since the existential quantifier is distributive over disjunction, and thus $\mathtt{TNF}(\varphi) \equiv \bigvee_i (\exists x \bigwedge_j (\psi^i_j))$. Since quantifiers are also distributive over formulas which do not contain the quantified variable, we get that the result, $\bigvee_i \Big(\bigwedge_{j \in J_i} \psi^i_j \land \exists x (\bigwedge_{j \notin J_i} \psi^i_j)\Big)$, is equivalent to $\exists x \mathtt{TNF}(\psi)$, and thus by inductive hypothesis also to φ. Since each formula ψ^i_j is, by inductive hypothesis, in the form τ, to show that the result is in TNF we only need to check that $\exists x (\bigwedge_{j \in J_i} \psi^i_j)$ is in the form τ. Syntactically this is trivial, and the constraint on variables in the TNF is indeed satisfied by the choice of J_i. The set of atoms does not increase by inductive hypothesis, and no new free variables appear by the choice of J_i. Furthermore, neither the quantifier rank nor the rank over any restricted variable increases. The case of universal quantification is analogous, modulo conversions between disjunctive and conjunctive normal forms (we assume that `CNF` and `DNF` do not create new atoms). $\square$

We will use the following important property of formulas in `TNF`.

Procedure TNF(φ)

```
case $\varphi$ is a literal return $\varphi$;
case $\varphi = \varphi_1 \lor \varphi_2$ return $\mathtt{TNF}(\varphi_1) \lor \mathtt{TNF}(\varphi_2)$;
case $\varphi = \varphi_1 \land \varphi_2$ return $\mathtt{TNF}(\varphi_1) \land \mathtt{TNF}(\varphi_2)$;
case $\varphi = \exists x \psi$ (or $\exists X \psi$) and $\mathtt{DNF}(\mathtt{TNF}(\psi)) = \bigvee_i (\bigwedge_j \psi^i_j)$
    Let $J_i = \{j \mid x \in \mathrm{free}(\psi^i_j)\}$; return $\bigvee_i \Big( \bigwedge_{j \notin J_i} \psi^i_j \land \exists x (\bigwedge_{j \in J_i} \psi^i_j)\Big)$;
case $\varphi = \forall x \psi$ (or $\forall X \psi$) and $\mathtt{CNF}(\mathtt{TNF}(\psi)) = \bigwedge_i (\bigvee_j \psi^i_j)$
    Let $J_i = \{j \mid x \in \mathrm{free}(\psi^i_j)\}$; return $\bigwedge_i \Big( \bigvee_{j \notin J_i} \psi^i_j \lor \forall x (\bigvee_{j \in J_i} \psi^i_j)\Big)$;
```

Lemma 9. *Let φ be a formula in TNF and $V_1, \ldots, V_n$ pairwise disjoint sets of variables such that if two variables appear in the same atom in φ, these variables belong to the same V_i. Then φ is a Boolean combination of formulas τ such that each τ contains only atoms with variables from one of the sets V_i.*

Proof. By contradiction, assume that there exists a formula φ in TNF which does not satisfy the above condition. Take such formula with smallest size (measured simply as the number of symbols). Then φ consists of only a single τ, since from a Boolean combination of more τ's one could choose a single one with atoms from different sets. Additionally, each sub-formula of φ satisfies the above lemma.

By assumption, $\varphi = \tau$ is not an atom, thus it is of the form $\exists X \mathcal{B}^+(\tau_i)$ or $\forall X \mathcal{B}^+(\tau_i)$ (or of the same form for first-order quantification). Each τ_i contains atoms only from a single set V_{j_i}, since otherwise it would be a smaller counter-example to the lemma and we have chosen τ as the smallest one. But, by the constraint on `TNF`, we know that X is contained in the free variables of each τ_i, and thus in each V_{j_i}. Since the sets V_i are pairwise disjoint, all j_i must be the same. This contradicts the assumption that τ contains atoms with variables from different sets V_i. $\square$

3.2 Formula Decomposition Algorithm

Let φ be a formula with only second-order free variables $X_1, \ldots, X_s$ and let $\mathcal{D}$ be a system of n structure equations

$$\mathcal{D} = \begin{cases} \Lambda^1 = \mathfrak{A}_1^1 \oplus \mathfrak{A}_2^1 \oplus \ldots \oplus \mathfrak{A}_{k_1}^1 & \text{with } f_1^1, \ldots, f_K^1 \\ \vdots \qquad\qquad \vdots & \qquad\qquad \vdots \\ \Lambda^n = \mathfrak{A}_1^n \oplus \mathfrak{A}_2^n \oplus \ldots \oplus \mathfrak{A}_{k_n}^n & \text{with } f_1^n, \ldots, f_K^n \end{cases}$$

with $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$. For each $m \in [n]$, the $\mathcal{D}_m$-decomposition of φ can be computed by performing the following steps:

(1) compute $\psi_m = \texttt{split}_{k_m}(\varphi)$;
(2) compute ϑ_m from ψ_m by replacing each atom $x^j \in X^k$ or $x^j = x^k$ with $\perp$ if $j \neq k$ and each atom $R_i(x_1^{j_1}, \ldots, x_{r_i}^{j_{r_i}})$ such that not all j_l are equal with $f_i^m(j_1, \ldots, j_{r_i})$;
(3) compute $\texttt{DNF(TNF(}\vartheta_m\texttt{))} = \bigvee_i \bigwedge_j \tau_{i,j}$.

We show that these steps indeed yield a $\mathcal{D}_m$-decomposition. By Lemma 5 and the definition of WMSO semantics we get that $\mathfrak{A}_m, \overline{P} \models \varphi \iff \mathfrak{A}_m, \overline{P^j} \models \psi_m$, where $P_i^j = P_i \cap \mathfrak{A}_m[j]$. Considering Step 2 of the algorithm, by the semantics of WMSO with restricted variables and the definition of $\mathcal{S}(\mathcal{D})$ we further get that $\mathfrak{A}_m, \overline{P^j} \models \psi_m \iff \mathfrak{A}_m, \overline{P^j} \models \vartheta_m$.

After this simplification step, all variables occurring in the same atomic sub-formula in ϑ_m are restricted to the same component, and by Lemma 9, each subformula $\tau_{i,j}$ in $\texttt{DNF(TNF(}\vartheta_m\texttt{))} = \bigvee_i \bigwedge_j \tau_{i,j}$ contains only atoms (and thus also quantifiers) with variables restricted to a single component. Let ψ_k^i be the conjunction of all $\tau_{i,j}$ containing variables restricted to the component $k \in [k_l]$, or $\top$ if no such $\tau_{i,j}$ occurs. Clearly $\texttt{TNF}(\vartheta_m)$ is equivalent to $\bigvee_i(\bigwedge_k \psi_k^i)$, and combining this with the previous equivalences we get that

$$\mathfrak{A}_m, \overline{P} \models \varphi \iff \mathfrak{A}_m, \overline{P}^j \models \bigvee_i (\bigwedge_k \psi_k^i).$$

To show that ψ_k^i with restricted variables X^k, x^k replaced by the standard ones X, x is a $\mathcal{D}_m$-decomposition of φ, it only remains to prove that $\mathrm{qr}(\tau_{i,j}) \leq \mathrm{qr}(\varphi)$ for all i, j. Observe that, by Lemma 5, we have $\mathrm{qr}^*(\psi_m) \leq \mathrm{qr}(\varphi)$. Replacing atoms does not change the quantifier rank, and by Theorem 8 we get that $\mathrm{qr}^*(\texttt{TNF}(\vartheta_m)) \leq \mathrm{qr}^*(\psi_m)$. But since each $\tau_{i,j}$ contains only quantification over variables from one component, we obtain that $\mathrm{qr}^*(\texttt{TNF}(\vartheta_m)) = \max_{i,j} \mathrm{qr}(\tau_{i,j}) \leq \mathrm{qr}(\varphi)$. This finally concludes the proof of Theorem 7.

4 Model Checking Algorithm

Our algorithm for model checking weak MSO sentences (i.e. formulas without free variables) on $\mathcal{S}_m(\mathcal{D})$ operates as follows.

- The only atomic sentences $\top$ and $\bot$ are verified trivially.
- Boolean combinations are verified by checking the subformulas and combining the results accordingly.
- Formulas of the form $\exists X \varphi(X)$ or $\exists x \varphi(x)$ are checked on $\mathcal{S}_m(\mathcal{D})$ by determining the winner of the finite reachability game $\mathcal{G}^\exists(\varphi, m)$ presented below.
- For formulas of the form $\forall X \varphi(X)$ or $\forall x \varphi(x)$ we check the equivalent formula $\neg\exists X \neg\varphi(X)$ or $\neg\exists x \neg\varphi(x)$, respectively, instead by determining the *loser* of the game $\mathcal{G}^\exists(\neg\varphi, m)$.

The main part of our model checking algorithm consists of establishing the winner of the following finite reachability game, which is based on the idea of decomposing formulas and on Theorem 7.

Definition 10. *Let $\exists X \varphi(X)$ be a sentence, $\Phi = \{\psi \mid \mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi), \mathrm{free}(\psi) \subseteq \{X\}\}$, and let $\mathcal{D}$ be a system of n structure equations. The two-player game $\mathcal{G}^\exists(\varphi, m)$ is played by the Verifier, who tries to show that $\mathcal{S}_m(\mathcal{D}) \models \exists X \varphi(X)$, against the Falsifier, who tries to disprove this. $\mathcal{G}^\exists(\varphi, m)$ is defined as follows.*

- *Positions of Verifier:* $\{\, [\psi, i] \mid \psi \in \Phi, i \in [n]\}$.
- *Positions of Falsifier:* $\{\, [(\psi_1, \ldots, \psi_{k_i}), S, i] \mid \psi_j \in \Phi, S \subseteq \bigcup_{j \in Fin_i} \mathfrak{A}_i[j]\}$.
- *Initial position:* $[\varphi, i]$.
- *Terminal positions:*

$$\{\, [\mathfrak{A}_j^i, \psi_j, S, i] \mid \lambda(i,j) = Fin, \psi_j \in \Phi\} \text{ and } \{\, [\varphi[X \leftarrow \emptyset], i] \mid \varphi \in \Phi, i \in [n]\}$$

– *Moves:* $[\varphi, i] \xrightarrow{V} [\varphi[X \leftarrow \emptyset], i]$,

$[\varphi, i] \xrightarrow{V} [(\psi_1, \ldots, \psi_{k_i}), S, i]$, *for each tuple* $(\psi_1, \ldots, \psi_{k_i})$ *in the* $\mathcal{D}_i$*-decomposition of* φ*, and*

$$[(\psi_1, \ldots, \psi_{k_i}), S, i] \xrightarrow{F} \begin{cases} [\mathfrak{A}^i_j, \psi_j, S, i] & \text{if } \lambda(i,j) = Fin \\ [\psi_j, \ell] & \text{if } \lambda(i,j) = \ell. \end{cases}$$

– *Winning condition: Verifier wins at a terminal position* $[\mathfrak{A}^i_j, \psi_j, S, i]$ *if and only if* $(\mathcal{S}_i(\mathcal{D})[j], S \cap \mathcal{S}_i(\mathcal{D})[j]) \models \psi_j(X)$*. At a position* $[\varphi[X \leftarrow \emptyset], i]$ *the Verifier wins if and only if* $\mathcal{S}_i(\mathcal{D}) \models \varphi[X \leftarrow \emptyset]$*. Falsifier wins infinite plays.*

Since the quantifier rank of the formulas in the decomposition tuples is bounded by the quantifier rank of φ and there are only finitely many non-equivalent formulas with fixed quantifier rank, Φ is finite. Furthermore, the size of the sets chosen by Verifier is bounded by the size of the structures in $\mathcal{D}$, and hence the arena of $\mathcal{G}^\exists(\varphi, m)$ is finite.

Theorem 11. *Verifier wins the game* $\mathcal{G}^\exists(\varphi, m)$ *if and only if* $\mathcal{S}_m(\mathcal{D}) \models \exists X \varphi(X)$*.*

Proof. We prove that there is a direct correspondence between winning strategies for Verifier and finite sets satisfying formulas.

($\Leftarrow$) Let $(\mathfrak{A}_1, \ldots, \mathfrak{A}_n) = \mathcal{S}(\mathcal{D})$ and assume that $\mathfrak{A}_m \models \exists X \varphi(X)$. Let S be a finite set such that $\mathfrak{A}_m, S \models \varphi$. We prove the existence of a winning strategy for Verifier by induction on the depth of S.

Let $\mathrm{dp}(S) = 1$, i.e. $S \subseteq \bigcup_{j \in \mathrm{Fin}_m} \mathfrak{A}_m[j]$. By Theorem 7 there exists a $\mathcal{D}_m$-decomposition $(\psi^1_1, \ldots, \psi^1_{k_m}), \ldots, (\psi^r_1, \ldots, \psi^r_{k_m})$ of φ and an index $\ell \in [r]$ such that $(\mathfrak{A}_m[j], S \cap \mathfrak{A}_m[j]) \models \psi^\ell_j$ for all $j \in [k_m]$. Since $\mathrm{dp}(S) = 1$, all elements in S are from the finite components of $\mathfrak{A}_m$, i.e. $S \cap \bigcup_{j \in \Delta_m} \mathfrak{A}_m[j] = \emptyset$, and $\mathfrak{A}_{\lambda(m,j)}, \emptyset \models \psi_j$ for all $j \in \Delta_m$. Hence, Verifier wins by moving to $[(\psi^\ell_1, \ldots, \psi^\ell_{k_m}), S, m]$: Falsifier cannot win by moving to a position $[\mathfrak{A}_m[j], \psi^\ell_j, S, m]$, for $j \in \mathrm{Fin}_m$, and from any position $[\psi^\ell_j]$, for $j \in \Delta_m$, Verifier can move to $[\psi^\ell_j[X \leftarrow \emptyset], \lambda(m, j)]$ and win.

Let $\mathrm{dp}(S) > 1$ and let $(\psi^1_1, \ldots, \psi^1_{k_m}), \ldots, (\psi^r_1, \ldots, \psi^r_{k_m})$ be the $\mathcal{D}_m$-decomposition of φ. Choose $\ell \in [r]$ such that $(\mathfrak{A}_m[j], S \cap \mathfrak{A}_m[j]) \models \psi^\ell_j$ for all $j \in [k_m]$. Let $S_0 = S \cap \bigcup_{j \in \mathrm{Fin}_m} \mathfrak{A}_m[j]$. We show that Verifier wins from $[(\psi^\ell_1, \ldots, \psi^\ell_{k_m}), S_0, m]$. If Falsifier chooses $j \in \mathrm{Fin}_m$ and moves to $[\mathfrak{A}_m[j], \psi^\ell_j, S_0, m]$, then Verifier wins because $(\mathfrak{A}_m[j], S \cap \mathfrak{A}_m[j]) \models \psi^\ell_j$. If Falsifiers chooses $j \in \Delta_m$, then we have that $\mathrm{dp}_j\left(\pi_j((S \setminus S_0) \cap \mathfrak{A}_m[j])\right) < \mathrm{dp}_j(S)$ (where $\pi_j : (s, jw) \mapsto (s, w)$), i.e. the depth of the remaining elements decreases upon descending into the j-th component. Since $(\mathfrak{A}_m[j], S_0 \cap \mathfrak{A}_m[j]) \models \psi^\ell_j$, applying the inductive hypothesis to positions $[\psi^\ell_j, \lambda(m, j)]$ for each $j \in \Delta_m$ we get that Verifier wins again.

($\Rightarrow$) Assume that Verifier has a strategy to win the game from the initial position $[\varphi(X), m]$. Since all plays won by Verifier are finite, unraveling the game graph and removing branches that do not correspond to moves taken by Verifier's winning strategy, we obtain a finite tree representing all possible plays of Falsifier against the fixed winning strategy of Verifier. The leaves of this tree are positions of the form $[\mathfrak{A}^i_j, \psi_j, S, i]$ or $[\psi[X \leftarrow \emptyset], i]$. We label the edges of the tree as follows: Edges representing Verifier's moves are labeled with ε; edges representing Falsifier's moves are labeled with letters from $\{1, \ldots, k_i\}$ corresponding to which part of the tuple Falsifier chooses, i.e. $[(\psi_1, \ldots, \psi_{k_i}), S, i] \xrightarrow{j} [\mathfrak{A}_i[j], \psi_j, S]$ or $[(\psi_1, \ldots, \psi_{k_i}), S, i] \xrightarrow{j} [\psi_j, \lambda(i,j)]$.

For each of Verifier's positions $p = [\psi, i]$ in the tree, we define the set $S(p)$ as the unique set which satisfies

$$S(p) \cap \mathfrak{A}_i[w] = S' \iff \text{a leaf } [\mathfrak{A}_w, \cdot, S', \cdot] \text{ is reachable from } p \text{ via labels } w$$

(note that the structure $\mathfrak{A}_w$ in the leaf, being one of the finite structures in $\mathcal{D}$, is actually isomorphic to $\mathfrak{A}_i[w]$). Intuitively, this set is obtained by combining all structures in reachable leaves after appropriately indexing their elements by the path w leading to them. We prove by induction on the *height* of positions in the tree that $\mathfrak{A}_i, S([\varphi, i]) \models \varphi$ holds for each position $[\varphi, i]$.

Let $\mathrm{h}([\varphi, i]) = 0$. Then the only successor is the leaf $[\varphi[X \leftarrow \emptyset], i]$, therefore $S([\varphi, i]) = \emptyset$ and by definition $(\mathfrak{A}_i, \emptyset) \models \varphi$.

Let $\mathrm{h}([\varphi, i]) > 0$. Then the only successor position $[(\psi_1, \ldots, \psi_{k_i}), S, i]$ has successors $[\mathfrak{A}_i[j], \psi_j, S'_j, i]$ (leaves), and $[\psi_j, \lambda(i,j)]$ with $\mathrm{h}([\psi_j, \lambda(i,j)]) < \mathrm{h}([\varphi, i])$. By induction hypothesis, $(\mathfrak{A}_{\lambda(i,j)}, S([\psi_j, \lambda(i,j)])) \models \psi_j$ for all $j \in \Delta_i$, and since we assume that Verifier plays a winning strategy, $(\mathfrak{A}_i[j], S'_j) \models \psi_j$ for $j \in \mathrm{Fin}_i$. Due to Theorem 7 we conclude that $(\mathfrak{A}_i, S([\varphi, i])) \models \varphi$. Considering the initial position $[\varphi, m]$ we obtain $(\mathfrak{A}_m, S([\varphi, m])) \models \varphi$, and hence $\mathfrak{A}_m \models \exists X \varphi(X)$. □

As presented, the model checking algorithm works in a top-down fashion and relies on solving finite reachability games. To establish the winner at positions of the form $[\psi[X \leftarrow \emptyset], j]$ in $\mathcal{G}^\exists(\varphi, i)$, we have to solve the model checking problem for the formula $\psi[X \leftarrow \emptyset]$, but note that $\psi[X \leftarrow \emptyset]$ has less variables and a smaller quantifier rank than $\exists X \varphi(X)$. Hence, the algorithm actually terminates.

Concerning the handling of existential first-order quantifiers there are two feasible approaches. By introducing a few special predicates for the subset relation and for expressing that a set is a singleton, one can avoid the use of first-order variables in the first place. On the other hand, the game can be easily modified to capture first-order quantification: Intuitively, instead of sets S, Verifier chooses either an element from one of the finite structures or announces in which of the inductively defined components the element is to be found.

5 Unbounding and Generalized Quantifiers

Many standard quantifiers, such as "there exists exactly one", do not increase the expressive power of MSO. One interesting exception is the unbounding quantifier: $UX\varphi$ expresses that the size of finite sets X satisfying φ is unbounded, i.e.

$$UX\varphi(X) \equiv \text{ for all } n \in \mathbb{N} \ \exists X \varphi(X) \text{ with } X \text{ finite and } |X| \geq n.$$

First introduced in [13], MSO with this quantifier was proven to be decidable on trees only with very restricted quantification patterns. Recently, only a technical analysis of max-automata allowed to show that satisfiability of WMSO with the unbounding quantifier is decidable on the class of all labelings of $(\omega, <)$ [12]. We prove that WMSO+U is decidable on all inductive structures, which is a more general result as far as the class of structures is concerned, but it is less general as we allow only finite labelings of the structures. For our proof, we only need to extend the algorithm presented above. Again, we fix a system $\mathcal{D}$ of n equations and let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$.

Definition 12. *A family $\mathcal{U} = \{S_i \mid i \in \mathbb{N}\}$ of finite sets is called* unbounded *in a component $\mathfrak{A}_m[j]$ if $\{i \mid \mathfrak{A}_m[j] \cap S_i \neq \emptyset\}$ is infinite.*

The following lemma is a consequence of the fact that our equations contain only a bounded number of structures.

Lemma 13. *Let $\mathcal{U} = \{S_i \mid (\mathfrak{A}_m, S_i) \models \varphi(X),\ |S_i| \geq i\}$ be a family of sets witnessing that $\mathfrak{A}_m \models UX\varphi(X)$. Then $\mathcal{U}$ is unbounded in some component $\mathfrak{A}_m[j]$.*

The above lemma, applied to k components, justifies the following extension of the $\texttt{split}_k$ procedure to the case $\varphi = UX\psi$ ($\overline{X}_{-j}$ denotes $\overline{X}$ without X^j):

$$\texttt{split}_k(\varphi) = \bigvee_{j=1,\ldots,k} \exists \overline{X^i}_{-j} UX^j \texttt{split}_k(\psi)[X \leftarrow \bigcup_i X^i].$$

The unbounding quantifier distributes over disjunctions, and the definition of $\texttt{TNF}$ and the conversion procedure for U is the same as for $\exists$. Thus, the theorem about $\mathcal{D}$-decompositions holds for WMSO+U as well.

To check WMSO+U, we proceed as for WMSO and instead of asking whether there exists a winning strategy, we impose different conditions on the set of all winning strategies of Verifier in the game.

Definition 14. *The game $\mathcal{G}^U(\varphi, m)$ is defined as $\mathcal{G}^\exists(\varphi, m)$ with only one addition: Falsifier's positions $[(\psi_1, \ldots, \psi_n), S, i]$ with $S \neq \emptyset$ are considered to be marked.*

By $\mathcal{T}_\sigma(\varphi, i)$ we denote the unraveling of the game graph from position $[\varphi, i]$ where all branches that are not chosen by Verifier's strategy σ are pruned.

Theorem 15. *$\mathfrak{A}_m \models UX\varphi(X)$ if and only if for each $n \in \mathbb{N}$, Verifier has a winning strategy σ_n such that $\mathcal{T}_{\sigma_n}(\varphi, m)$ contains at least n marked positions.*

Proof. ($\Rightarrow$) Let M be the maximum number of elements in the universe of all finite structures appearing in $\mathcal{D}$ and assume that $\mathfrak{A}_m \models UX\varphi(X)$. Thus, for each $n \in \mathbb{N}$ there is a set S_n with $|S_n| \geq n$ such that $\mathfrak{A}_m, S_n \models \varphi(X)$. Following the same arguments as in the proof of Theorem 11, each S_n gives rise to a winning strategy σ_n for Verifier, namely "choose the upcoming elements of S_n." Consider the strategy $\sigma_{n \cdot M}$. Since $\sigma_{n \cdot M}$ chooses elements from $S_{n \cdot M}$, and at each marked position at most M of those, it follows from $|S_{n \cdot M}| \geq n \cdot M$ that there are at least n marked positions in $\mathcal{T}_{\sigma_{n \cdot M}}(\varphi, m)$.

($\Leftarrow$) Given a winning strategy σ, we construct, as in the proof of Theorem 11, a set S_σ satisfying φ. Consider a strategy σ_n with at least n marked positions in $\mathcal{T}_{\sigma_n}(\varphi, m)$. Since each marked position corresponds to a choice of a non-empty subset, and these subsets are disjoint, $|S_{\sigma_n}| \geq n$. Hence, $\mathfrak{A}_m \models UX\varphi(X)$ as we have assumed the existence of a winning strategy for each $n \in \mathbb{N}$. $\square$

For a reachability game with a finite arena, the above condition, i.e. the existence of winning strategies which result in game trees containing arbitrarily many marked positions, can be verified by a basic graph algorithm. Including any such procedure into our model checking algorithm, we obtain a procedure for model checking WMSO+U formulas on arbitrary inductive structures.

6 Implementation

We implemented a prototype in OCaml interfacing to MiniSat for performing CNF $\leftrightarrow$ DNF conversions following the idea described in [14]. The implementation[1] is functional but still leaves much room for improvement and optimization.

For a comparison with Mona we ran two tests—checking simple formulas of Presburger arithmetic taken from the examples shipped with Mona, and artificially constructed Horn formulas of the form

$$\varphi_n := \exists X \forall x_1 \dots \forall x_n \big((x_1 \in X \to x_2 \in X) \wedge \dots \wedge (x_{n-1} \in X \to x_n \in X) \big) .$$

The results in Table 1 show that Presburger arithmetic presents no problem for Mona since an automaton recognizing addition is fairly small and easy to construct. For the prototype, the result depends on whether the constants are encoded in the input formula (A) or in the structure equations (B). On the other hand, the Horn formulas could be easily decomposed by our algorithm whereas Mona soon reaches its limits, being only able to handle formulas up to φ_{15}. This supports our claim that there are verification problems that might be better suited for a treatment on a logical level while there are others for which automata theoretic approaches are adequate.

However, due to the lack of example formulas, not to mention a benchmark suite, and the evident need for further optimization of our prototype, it is hard to carry out a meaningful comparison.

	Prototype A	B	Mona
$\exists x(2x = 9)$	0.5	0.1	0.1
$\exists x(2x = 16)$	3	0.6	0.1
$\exists x(2x = 24)$	8	0.6	0.1
$\exists x(2x = 25)$	7	0.1	0.1

	Prototype	Mona
φ_{14}	0.1	7
φ_{15}	0.1	17
φ_{100}	0.3	–
φ_{500}	12	–

Table 1. Comparison of the running times measured in seconds

[1] Available from `toss.sourceforge.net`, SVN revision 1049, in Solver/

7 Future Work

Unlike advances in complementation and minimization techniques for automata, which usually do not provide any new intuitions about the logical aspects of the model-checking procedure, we think that, in addition to the pure algorithmic value, our method can provide new insights into the composition method and might help to understand the algebraic structure of tree languages definable in weak MSO. Moreover, we aim at extending our method to further logics. Similar to the presented modification of the game that yields a decision procedure for WMSO+U, the game might be extended to capture other quantifiers. Additionally, we hope that our method can at least partially be extended to richer fragments of MSO and, as a long term goal, give an insight into the structure of tree languages definable in various fragments of MSO.

References

1. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: International Congress on Logic, Methodology and Philosophy of Science, Stanford University Press (1962) 1–11
2. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Transactions of the American Mathematical Society **141** (1969) 1–35
3. Courcelle, B.: The monadic second order logic of graphs, II: Infinite graphs of bounded width. Mathematical System Theory **21** (1989) 187–222
4. Basin, D.A., Klarlund, N.: Hardware verification using monadic second-order logic. In: Proceedings of CAV '95. Volume 939 of LNCS., Springer (1995) 31–41
5. Jensen, J.L., Jørgensen, M.E., Klarlund, N., Schwartzbach, M.I.: Automatic verification of pointer programs using monadic second-order logic. In: Proceedings of PLDI '97. (1997) 226–236
6. Zee, K., Kuncak, V., Rinard, M.C.: An integrated proof language for imperative programs. In: PLDI. (2009) 338–351
7. Shelah, S.: The monadic second order theory of order. Annals of Mathematics **102** (1975) 379–419
8. Thomas, W.: Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In: Structures in Logic and Computer Science. Volume 1261 of LNCS. Springer-Verlag (1997) 118–143
9. Läuchli, H.: A decision procedure for the weak second order theory of linear order. In H. Arnold Schmidt, K.S., Thiele, H.J., eds.: Proceedings of the Logic Colloquium 1966. Volume 50. Elsevier (1968) 189–197
10. Blumensath, A., Colcombet, T., Löding, C.: Logical theories and compatible operations. In Flum, J., Grädel, E., Wilke, T., eds.: Logic and automata: History and Perspectives. Amsterdam University Press (2007) 72–106
11. Makowsky, J.A.: Algorithmic uses of the Feferman-Vaught theorem. Annals of Pure and Applied Logic **126.1-3** (2004) 159–213
12. Bojanczyk, M.: Weak MSO with the unbounding quantifier. In: Proceedings of STACS '09. Volume 09001 of LIPIcs., Schloss Dagstuhl (IBFI) (2009) 159–170
13. Bojanczyk, M.: A bounding quantifier. In: Proceedings of CSL '04. Volume 3210 of LNCS., Springer (2004) 41–55
14. McMillan, K.L.: Applying sat methods in unbounded symbolic model checking. In: Proceedings of CAV 2002. (2002) 250–264

Appendices

Note that throughout the whole appendix, we also allow free first-order variables in formulas, and hence the definitions and notations differ from those used in the main part of the paper.

Appendix A provides additional details concerning the formal semantics of formulas with restricted variables and the rather technical proof of Lemma 5. Appendix B refines the notion of a decomposition accounting for free first-order variables which yields the basis for a game capturing first-order quantification described in Appendix C, and thus completes the model-checking algorithm described in Section 4.

A Formulas with Restricted Variables

A.1 Semantics

We formally define the semantics of τ-formulas with restricted variables of k kinds by a translation into formulas over the expanded vocabulary $\hat{\tau} = \tau \cup \{P^1, \ldots, P^k\}$ where P^i are unary predicates not contained in τ. Given a formula φ with restricted variables, let $\hat{\varphi}$ be the formula obtained from φ by replacing

$$\begin{aligned}
x^i = y^j &\quad\rightsquigarrow\quad x \in P^i \wedge y \in P^j \wedge x = y \\
R(x_1^{i_1}, \ldots, x_r^{i_r}) &\quad\rightsquigarrow\quad \Big(\bigwedge_{j=1,\ldots,r} x_j \in P^{i_j}\Big) \wedge R(x_1, \ldots, x_r), \\
x^i \in Y^j &\quad\rightsquigarrow\quad x \in P^i \wedge x \in P^j \wedge x \in Y \\
\forall x^i \varphi(x^i) &\quad\rightsquigarrow\quad \forall x_i (x_i \in P^i \to \varphi(x_i)) \qquad [x_i \text{ is a fresh variable}] \\
\exists x^i \varphi(x^i) &\quad\rightsquigarrow\quad \exists x_i (x_i \in P^i \wedge \varphi(x_i)) \\
\forall X^i \varphi(X^i) &\quad\rightsquigarrow\quad \forall X_i (X_i \subseteq P^i \to \varphi(X_i)) \\
\exists X^i \varphi(X^i) &\quad\rightsquigarrow\quad \exists X_i (X_i \subseteq P^i \wedge \varphi(X_i)).
\end{aligned}$$

Note that the first three items are mainly important if the formula contains free variables since the range of quantified variables is already appropriately restricted by the guards. Given a τ-structure $\mathfrak{A}$ and a partition of its universe into k sets $A^1, \ldots, A^k$, we refer to $\mathfrak{A}_{\langle A^1,\ldots,A^k\rangle}$ as the *partitioned structure*, and denote the $\hat{\tau}$-expansion of $\mathfrak{A}$ in which each P^i is interpreted as the set A^i as usual by $(\mathfrak{A}, A^1, \ldots, A^k)$. The semantics of φ evaluated on a partitioned τ-structure given an assignment β of the free first- and second-order variables is defined by $(\mathfrak{A}_{\langle A^1,\ldots,A^k\rangle}, \beta) \models \varphi$ if and only if $(\mathfrak{A}, A^1, \ldots, A^k, \beta) \models \hat{\varphi}$. (Note that β is an assignment of the free original variables, and not of each restricted occurrence!)

A.2 Splitting Variables

The following extended procedure $\texttt{split}_k$ also handles free first-order variables. Note that it is important that the replacement of the free variables is done first

```
Procedure split_k(φ)
    case $\varphi$ contains a free (unrestr.) FO-var. $x$ return $\mathtt{split}_k(\bigvee_{i=1,\dots,k} \varphi[x \leftarrow x^i])$;
    case $\varphi$ contains a free (unrestr.) MSO-var. $X$ return $\mathtt{split}_k(\varphi[X \leftarrow \bigcup_i X^i])$;
    case $\varphi$ is an atom return $\varphi$;
    case $\varphi = \neg\psi$ return $\neg\mathtt{split}_k(\psi)$;
    case $\varphi = \varphi_1 \vee \varphi_2$ return $\mathtt{split}_k(\varphi_1) \vee \mathtt{split}_k(\varphi_2)$;
    case $\varphi = \varphi_1 \wedge \varphi_2$ return $\mathtt{split}_k(\varphi_1) \wedge \mathtt{split}_k(\varphi_2)$;
    case $\varphi = \exists x\psi$ return $\bigvee_{i=1,\dots,k} \exists x^i \mathtt{split}_k(\psi)[x \leftarrow x^i]$;
    case $\varphi = \forall x\psi$ return $\bigwedge_{i=1,\dots,k} \forall x^i \mathtt{split}_k(\psi)[x \leftarrow x^i]$;
    case $\varphi = \exists X\psi$ return $\exists X^1 \dots X^k \mathtt{split}_k(\psi)[X \leftarrow \bigcup_i X^i]$;
    case $\varphi = \forall X\psi$ return $\forall X^1 \dots X^k \mathtt{split}_k(\psi)[X \leftarrow \bigcup_i X^i]$;
```

before splitting the rest of the formula. We obtain the following modified version of the splitting lemma.

Lemma 16. *For every structure $\mathfrak{A}$ every partition $(A^1, \dots, A^k)$ of the universe of $\mathfrak{A}$ and every assignment β of the free variables occurring in φ, it holds that $\mathfrak{A}, \beta \models \varphi$ if and only if $\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta \models \mathtt{split}_k(\varphi)$. Moreover, $\mathrm{qr}^*(\mathtt{split}_k(\varphi)) \leq \mathrm{qr}(\varphi)$.*

Proof. We show the equivalence of the split formula by induction on the structure of formulas.
Atomic formulas:

– $\varphi = (x = y)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models x = y \iff \beta(x) = \beta(y)\\
\iff &\text{ex. } i, j \in [k] \text{ such that } \beta(x) \in A^i,\ \beta(y) \in A^j, \text{ and } \beta(x) = \beta(y)\\
\iff &(\mathfrak{A}, A^1, \dots, A^k, \beta) \models \bigvee_{i=1,\dots,k} \bigvee_{j=1,\dots,k} \underbrace{x \in P^i \wedge y \in P^j \wedge x = y}_{\text{translation of } x^i = y^j}\\
\iff &(\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta) \models \bigvee_{i=1,\dots,k} \bigvee_{j=1,\dots,k} x^i = y^j = \mathtt{split}_k(\varphi)
\end{aligned}$$

– $\varphi = R(x_1, \dots, x_r)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models R(x_1, \dots, x_r)\\
\iff &(\beta(x_1), \dots, \beta(x_r)) \in R^{\mathfrak{A}}\\
\iff &\text{ex. } i_1, \dots, i_r \text{ such that}\\
&\beta(x_1) \in A^{i_1}, \dots, \beta(x_r) \in A^{i_r}, \text{ and } (\beta(x_1), \dots, \beta(x_r)) \in R^{\mathfrak{A}}\\
\iff &(\mathfrak{A}, A^1, \dots, A^k, \beta) \models \bigvee_{(i_1,\dots,i_r)\in[k]^r} \underbrace{\bigwedge_{j=1,\dots,r} x_j \in P^{i_j} \wedge R(x_1, \dots, x_r)}_{\text{translation of } R(x_1^{i_1}, \dots, x_r^{i_r})}\\
\iff &(\mathfrak{A}_{\langle A^1,\dots,A^k\rangle}, \beta) \models \bigvee_{(i_1,\dots,i_r)\in[k]^r} R(x_1^{i_1}, \dots, x_r^{i_r}) = \mathtt{split}_k(\varphi)
\end{aligned}$$

– $\varphi = x \in Y$

$$\begin{aligned}
&\mathfrak{A}, \beta \models x \in Y \\
\iff &\text{ex. } i \in [k] \text{ such that } \beta(x) \in A^i, \text{ and } \beta(x) \in \beta(Y) \\
\iff &\text{ex. } i \in [k] \text{ such that } \beta(x) \in A^i, \text{ and } \beta(x) \in \bigcup_j (\beta(Y) \cap A^j) \\
\iff &(\mathfrak{A}, A^1, \ldots, A^k, \beta) \models \bigvee_{i=1,\ldots,k} \bigvee_{j=1,\ldots,k} \underbrace{x \in P^i \wedge x \in P^j \wedge x \in Y}_{\text{translation of } x^i \in Y^j} \\
\iff &(\mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta) \models \bigvee_{i=1,\ldots,k} \bigvee_{j=1,\ldots,k} x^i \in Y^j = \mathtt{split}_k(\varphi)
\end{aligned}$$

Inductive step:

– If φ is a Boolean combination, the statement is obvious.
– $\varphi = \exists x \psi(x)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models \exists x \psi(x) \\
\iff &\text{ex. } a \in A \text{ such that } \mathfrak{A}, \beta[x \mapsto a] \models \psi(x) \\
\iff &\text{ex. } i \text{ and } a \in A^i \text{ such that } \mathfrak{A}, \beta[x \mapsto a] \models \psi(x) \\
\iff &\text{ex. } i \text{ and } a \in A^i \text{ such that } \mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta[x \mapsto a] \models \mathtt{split}_k \psi(x) \\
\iff &\mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta \models \bigvee_{i=1,\ldots,k} \exists x^i \mathtt{split}_k \psi(x^i)
\end{aligned}$$

– $\varphi = \forall x \psi(x)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models \forall x \psi(x) \\
\iff &\text{for all } a \in A, \text{ we have } \mathfrak{A}, \beta[x \mapsto a] \models \psi(x) \\
\iff &\text{for all } i \text{ and } a \in A^i, \text{ we have } \mathfrak{A}, \beta[x \mapsto a] \models \psi(x) \\
\iff &\text{for all } i \text{ and } a \in A^i, \text{ we have } \mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta[x \mapsto a] \models \mathtt{split}_k \psi(x) \\
\iff &\mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta \models \bigwedge_{i=1,\ldots,k} \forall x^i \mathtt{split}_k \psi(x^i)
\end{aligned}$$

– $\varphi = \exists X \psi(X)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models \exists X \psi(X) \\
\iff &\text{ex. } S \subseteq A \text{ such that } \mathfrak{A}, \beta[X \mapsto S] \models \psi(X) \\
\iff &\text{ex. } S_1 \subseteq A^1, \ldots, S_k \subseteq A^k \text{ such that } \mathfrak{A}, \beta[X_i \mapsto S_i] \models \psi[X \leftarrow \cup X_i] \\
\iff &\text{ex. } S_1 \subseteq A^1, \ldots, S_k \subseteq A^k \\
&\text{such that } \mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta[X_i \mapsto S_i] \models \mathtt{split}_k(\psi[X \leftarrow \cup X_i]) \\
\iff &\mathfrak{A}_{\langle A^1,\ldots,A^k \rangle}, \beta \models \exists X^1 \ldots X^k \mathtt{split}_k(\psi[X \leftarrow \cup X^i])
\end{aligned}$$

– $\varphi = \forall X \psi(X)$

$$\begin{aligned}
&\mathfrak{A}, \beta \models \forall X \psi(X) \\
\Longleftrightarrow\ &\text{for all } S \subseteq A, \text{ we have } \mathfrak{A}, \beta[X \mapsto S] \models \psi(X) \\
\Longleftrightarrow\ &\text{for all } S_1 \subseteq A^1, \dots, S_k \subseteq A^k, \text{ we have } \mathfrak{A}, \beta[X_i \mapsto S_i] \models \psi[X \leftarrow \cup X_i] \\
\Longleftrightarrow\ &\text{for all } S_1 \subseteq A^1, \dots, S_k \subseteq A^k, \\
&\text{we have } \mathfrak{A}_{\langle A^1, \dots, A^k \rangle}, \beta[X_i \mapsto S_i] \models \texttt{split}_k(\psi[X \leftarrow \cup X_i]) \\
\Longleftrightarrow\ &\mathfrak{A}_{\langle A^1, \dots, A^k \rangle}, \beta \models \forall X^1 \dots X^k \texttt{split}_k(\psi[X \leftarrow \cup X^i]) \qquad \square
\end{aligned}$$

B Decomposing Formulas with free FO-variables

Unlike in the case of free second-order variables, the notion of a decomposition gets more complicated in the presence of free first-order variables since a variable assignment of a first-order variable by an element from, say, the i-th component of the structure cannot meaningfully interpret the restriction of the variable to another component. However, by an adequate modification of the original definition of a decomposition, we show that basically the same algorithm also works for computing decompositions of formulas with free first-order variables.

Definition 17. *Let $\mathcal{D}$ be a system of n structure equations such that k_i structures appear on the right-hand side of the i-th equation. Let $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$ and let φ be a WMSO formula with free variables $\mathrm{free}(\varphi) = \{x_1, \dots, x_r, X_1, \dots, X_s\}$. For each $m \in [n]$, a $\mathcal{D}_m$-*decomposition *of φ is a sequence of k-tuples ($k = k_m$) of formulas $(\psi_1^1, \dots, \psi_k^1), \dots, (\psi_1^l, \dots, \psi_k^l)$ such that the free variables of each ψ_j^i are included in $\mathrm{free}(\varphi)$, $\mathrm{qr}(\psi_j^i) \leq \mathrm{qr}(\varphi)$, and*

$$\mathfrak{A}_m, \beta \models \varphi \iff \textit{for some } i \in [l] \textit{ and each } j \in [k] \quad \mathfrak{A}_m[j], \hat{\beta}_j \models \hat{\psi}_j^i.$$

where β is an assignment of the free first- and second-order variables, and $\hat{\beta}_j$ is a partial assignment defined as follows:

- *$\hat{\beta}_j$ is undefined for variables x such that $\beta(x) \notin A_m[j]$,*
- *$\hat{\beta}_j(x) = \beta(x)$ otherwise, and*
- *$\hat{\beta}_j(X) = \beta(X) \cap \mathfrak{A}_m[j]$.*

Furthermore,

$$\hat{\psi}_j^i = \begin{cases} \bot & \textit{if } \psi_j^i \textit{ contains a free variable } x \textit{ with } \beta(x) \notin A_m[j] \\ \psi_j^i & \textit{otherwise .} \end{cases}$$

The modification of the formulas ensures that all free first-order variables in the ψ_j^i are actually interpreted by $\hat{\beta}_j$ and captures the intuition that a first-order variable cannot be assigned a value from different components at the same time.

Again our aim is to prove

Theorem 18. *For every WMSO formula φ (with arbitrary free variables), every system of n structure equations $\mathcal{D}$, and $m \in [n]$, there exists an effectively computable $\mathcal{D}_m$-decomposition of φ.*

Towards this goal, we show that the same algorithm as presented in Section 3.2 also yields a decomposition in the above sense, which only differs from Definition 6 in presence of free FO-variables.

In the following, for a formula φ with free first-order variables $\{x_1, \dots, x_r\}$ and a tuple $(c_1, \dots, c_r)$, we will use the notation $\varphi^{\bar{c}}$ to denote the formula $\varphi[x_1 \leftarrow x_1^{c_1}, \dots, x_r \leftarrow x_r^{c_r}])$. Furthermore, given an assignment β of the free first-order variables, we denote by $\bar{c}_\beta = (c_1, \dots, c_r)$ the tuple of components such that $\beta(x_i) \in A^{c_i}$ for $i = 1, \dots, r$.

The first step of the algorithm computes $\psi_m = \mathtt{split}_{k_m}(\varphi)$. Assume that φ contains the free variables $x_1, \dots, x_r$. Then, after splitting all first-order variables, we obtain

$$\psi_m = \bigvee_{(c_1,\dots,c_r)\in[k_m]^r} \underbrace{\mathtt{split}_{k_m}(\varphi^{\bar{c}})}_{\eta_m^{\bar{c}}},$$

i.e. ψ_m contains, for every possible combination of components, a disjunct in which each free variable is restricted to that specific component. Furthermore, by Lemma 16, $\mathfrak{A}, \beta \models \varphi \iff \mathfrak{A}_{\langle A^1,\dots,A^{k_m}\rangle}, \beta \models \mathtt{split}_{k_m}(\varphi)$.

By the definition of the semantics in A.1, it is clear that a formula containing a variable x_j restricted to c_j is evaluated to $\perp$ under an assignment β with $\beta(x_j) \notin A^{c_j}$. Hence, for every assignment β, there is exactly one disjunct in ψ_m that does not default to $\perp$, and by the previous observation, we get in fact, $\mathfrak{A}, \beta \models \varphi \iff \mathfrak{A}_{\langle A^1,\dots,A^{k_m}\rangle}, \beta \models \mathtt{split}_{k_m}(\varphi^{\bar{c}_\beta})$.

Since the definition of the TNF does not impose constraints on occurrences of free variables, the transformation $\mathtt{TNF}(\varphi(\overline{x}))$ indeed yields a formula with the same free variables in TNF. Furthermore, by definition of the procedure transforming formulas into TNF, $\mathtt{TNF}(\psi_m) = \mathtt{TNF}(\bigvee_{\bar{c}} \eta_m^{\bar{c}}) = \bigvee_{\bar{c}} \mathtt{TNF}(\eta_m^{\bar{c}})$, and due to commutativity of disjunction also $\mathtt{DNF}(\mathtt{TNF}(\bigvee_{\bar{c}} \eta_m^{\bar{c}})) = \bigvee_{\bar{c}} \mathtt{DNF}(\mathtt{TNF}(\eta_m^{\bar{c}}))$.

As each disjunct represents a tuple of formulas in the sequence which we claim to be a $\mathcal{D}_m$-decomposition, this sequence can be seen as a concatenation of subsequences—one for each formula $\varphi^{\bar{c}}$.

By the definition of the semantics, it is clear that $\mathfrak{A}, \beta \models \varphi$ if and only if $\mathfrak{A}, \beta \models \varphi^{c_\beta}$. Let β be fixed, and assume that $\mathfrak{A}_m, \beta \models \varphi$. By the same reasoning as in Section 3.2, in the subsequence obtained from the decomposition of $\varphi^{\bar{c}_\beta}$ there is a tuple $(\psi_1, \dots, \psi_{k_m})$ such that $\mathfrak{A}_m[j], \hat{\beta}^j \models \hat{\psi}_j$ for each $j = 1, \dots, k_m$. (Note that in this case, $\hat{\beta}_j(x_i) = \beta(x_i)$ and $\hat{\psi}_j = \psi_j$ for all $j \in [k_m]$.) On the other hand, assume there is a tuple $(\psi_1, \dots, \psi_{k_m})$ such that $\mathfrak{A}_m[j], \hat{\beta}^j \models \hat{\psi}_j$ for each $j = 1, \dots, k_m$, then this tuple must originate from the decomposition of $\varphi^{\bar{c}_\beta}$. Otherwise, $\hat{\psi}_j = \perp$ for some j, contradicting the assumption. Hence, by the semantical equivalence of the TNF and the splitting lemma, $\mathfrak{A}_m, \beta \models \varphi^{\bar{c}_\beta}$, and by the above observation also $\mathfrak{A}_m, \beta \models \varphi$.

Example 19. Consider the formula $\varphi(x) = \forall y(x = y \vee x < y)$ on the binary tree as defined in Example 4. The decomposition is computed by first splitting the formula:

$$\begin{aligned}\psi_1(\varphi) &= \bigvee_{i=1}^{3} \mathtt{split}_3(\forall y(x^i = y \vee x^i < y)) \\ &= \bigvee_{i=1}^{3} \bigwedge_{j=1}^{3} \forall y^j (x^i = y^j \vee x^i < y^j) \ ,\end{aligned}$$

and in the following simplification step, replacing atoms containing variables from different components, the formula is reduced to $\forall y^1(x^1 = y^1 \vee x^1 < y^1)$. Hence, the decomposition is the singleton sequence $\big(\forall y(x = y \vee x < y), \top, \top\big)$.

This reflects the fact that, in order to fulfill the formula, the variable x must be assigned an element from the first component of the structure (which leaves the root node of the tree as the only possibility). Indeed, evaluating the formula on the first component of the tree (i.e., the structure consisting of only the root node) shows that $\exists x\varphi(x)$ holds in the binary tree.

C First-Order existential Game

Definition 20. *Let $\exists x\varphi(x, \bar{y})$ be a formula, and β an assignment of all free variables $\bar{y}$. Let $\Phi = \{\psi \mid \mathrm{qr}(\psi) \leq \mathrm{qr}(\varphi), \mathrm{free}(\psi) \subseteq \{x, \bar{y}\}\}$, and let $\mathcal{D}$ be a system of n structure equations. The two-player game $\mathcal{G}^{\exists_1}(\varphi, \beta, m)$ is played by the Verifier, who tries to show that $\mathcal{S}_m(\mathcal{D}), \beta \models \exists x\varphi(x, \bar{y})$, against the Falsifier, who tries to disprove this. $\mathcal{G}^{\exists_1}(\varphi, \beta, m)$ is defined as follows.*

- *Positions of Verifier:* $\{\,[\psi, \beta, i] \mid \psi \in \Phi, i \in [n]\}$.
- *Positions of Falsifier:* $\{\,[(\psi_1, \dots, \psi_{k_i}), \beta, c, i] \mid \psi_j \in \Phi, c \in [k_i]\}$.
- *Initial position:* $[\varphi, \beta, i]$.
- *Terminal positions:*

$$\begin{aligned}&\{\,[\mathfrak{A}^i_j, \psi_j, \beta, c, i] \mid j \neq c,\ \psi_j \in \Phi\} \\ &\{\,[\mathfrak{A}^i_j, \psi_j, \beta, c, i] \mid \lambda(i,j) = \mathit{Fin},\ \psi_j \in \Phi\} \text{ and} \\ &\{\,[\varphi, \beta[x \mapsto a], i] \mid \varphi \in \Phi,\ i \in [n], a \in \bigcup_{j \in \mathit{Fin}_i} \mathfrak{A}_i[j]\}\end{aligned}$$

- *Moves:*

$$\begin{aligned}&[\varphi, \beta, i] \xrightarrow{V} [\varphi, \beta[x \mapsto a], i] \\ &[\varphi, \beta, i] \xrightarrow{V} [(\psi_1, \dots, \psi_{k_i}), \beta, c, i], \textit{ for each tuple } (\psi_1, \dots, \psi_{k_i}) \\ &\qquad\qquad \textit{in the } \mathcal{D}_i\textit{-decomposition of } \varphi, \textit{ and}\end{aligned}$$

$$[(\psi_1, \dots, \psi_{k_i}), \beta, c, i] \xrightarrow{F} \begin{cases} [\mathfrak{A}^i_j, \psi_j, \beta, c, i] & \textit{if } \lambda(i,j) = \mathit{Fin} \\ [\psi_j, \beta, \ell] & \textit{if } \lambda(i,j) = \ell. \end{cases}$$

- *Winning condition: Falsifier wins at a terminal position $[\mathfrak{A}^i_j, \psi_j, \beta, c, i]$ if $x \in \mathrm{free}(\psi_j)$. If $x \notin \mathrm{free}(\psi_j)$, then Verifier wins if and only if $\mathfrak{A}^i_j, \beta \models \psi_j$. At a position $[\varphi, \beta[x \mapsto a], i]$ Verifier wins if and only if $\mathcal{S}_i(\mathcal{D}), \beta[x \mapsto a] \models \varphi(x)$. Infinite plays are won by Falsifier.*

Since the quantifier rank of the formulas in the decomposition tuples is bounded by the quantifier rank of φ and there are only finitely many non-equivalent formulas with fixed quantifier rank, Φ is finite. Furthermore, there are only finitely many possible elements for Verifier to choose, and hence the arena of $\mathcal{G}^{\exists_1}(\varphi, \beta, m)$ is finite.

Theorem 21. *Verifier wins the game $\mathcal{G}^{\exists_1}(\varphi, \beta, m)$ if and only if $\mathcal{S}_m(\mathcal{D}), \beta \models \exists x\varphi(x, \bar{y})$.*

Proof. We prove that there is a direct correspondence between winning strategies for Verifier and finite sets satisfying formulas.

($\Leftarrow$) Let $(\mathfrak{A}_1, \ldots, \mathfrak{A}_n) = \mathcal{S}(\mathcal{D})$ and assume that $\mathfrak{A}_m, \beta \models \exists x\varphi(x, \bar{y})$. We can assume that $\mathfrak{A}_m$ is an indexed structure, and we let a be the indexed element such that $\mathfrak{A}_m, \beta[x \mapsto a] \models \varphi(x, \bar{y})$. We prove the existence of a winning strategy for Verifier by induction on the depth of $a = (a_0, w)$. Let $\mathrm{dp}(a) = 1$, then a is an element in a finite component of $\mathfrak{A}_m$, and hence Verifier wins by moving to $[\varphi, \beta[x \mapsto a_0], m]$. Let $\mathrm{dp}(a) > 1$, i.e. $a = (a_0, cw')$. By Theorem 18 there exists a $\mathcal{D}_m$-decomposition $(\psi_1^1, \ldots, \psi_{k_m}^1), \ldots, (\psi_1^r, \ldots, \psi_{k_m}^r)$ of φ and a number $\ell \in [r]$ such that $(\mathfrak{A}_m[j], \hat{\beta}_j[x \mapsto a]) \models \hat{\psi}_j^\ell$ for all $j \in [k_m]$. Since a is to be found in the c-th component (which cannot be a finite one), ψ_c^ℓ is the only formula of the tuple in which x occurs freely (otherwise, $\hat{\psi}_c^\ell = \bot$), and hence, by induction hypothesis, Verifier wins by moving to $[(\psi_1^\ell, \ldots, \psi_{k_m}^\ell), c, m]$.

($\Rightarrow$) Assume that Verifier has a strategy to win the game from the initial position $[\varphi(x, \bar{y}), \beta, m]$. Since all plays won by Verifier are finite, unraveling the game graph and removing branches that do not correspond to moves taken by Verifier's winning strategy, and Falsifier's moves that directly lead to terminal positions that are losing for him, we obtain a finite path (since Falsifier has only trivial choices if he does not want to lose immediately). Concatenating Verifier's choices of components along this path to an index word w, we obtain, together with the final move to position $[\varphi, \beta[x \mapsto a], i]$, an indexed element (a, w). By a straightforward induction using Theorem 18, we conclude, that $\mathfrak{A}_m, \beta[x \mapsto a] \models \varphi(x, \bar{y})$. □

D Proof from Section 5

Proof (of Lemma 13). For each $i \in \mathbb{N}$, let S_i be a set with more than i elements satisfying $\varphi(S_i)$. The tree obtained from the prefix closure of $\{w : (s, w) \in \bigcup_i S_i\}$ (w is the index of the element w.r.t. $\mathfrak{A}_m$) is a finitely branching tree with arbitrarily long paths (since the depth of the sets S_i is unbounded) such that, for each node v, the component $\mathcal{S}_m(\mathcal{D})[v]$ contains some element from a set S_i. By König's Lemma, this tree has an infinite path $\alpha = j\alpha_1 \in \Delta_{\mathcal{D}}^\omega$, with $\Delta_{\mathcal{D}} = \bigcup_\ell \Delta_\ell$, and since each S_i is finite, elements of infinitely many S_i are reachable from any node on this path. In particular, $\mathcal{U}$ is unbounded in $\mathcal{S}_m(\mathcal{D})[j]$. In general, for any $\mathcal{S}_m(\mathcal{D})[w]$, where w is a prefix of $\alpha = wj\alpha'$, $\mathcal{U}$ is unbounded in the component $\mathcal{S}_m(\mathcal{D})[wj]$. □