Linear-Time Algorithms for Graphs of Bounded Rankwidth: A Fresh Look Using Game Theory[☆,☆☆]

Alexander Langer, Peter Rossmanith, Somnath Sikdar

Department of Computer Science, RWTH Aachen University, 52074 Aachen, Germany

Abstract

We present an alternative proof of a theorem by Courcelle, Makowski and Rotics [7] which states that problems expressible in MSO_1 are solvable in linear time for graphs of bounded rankwidth. Our proof uses a game-theoretic approach and has the advantage of being self-contained. In particular, our presentation does not assume any background in logic or automata theory. We believe that it is good to have alternative proofs of this important result. Moreover our approach can be generalized to prove other results of a similar flavor, for example, that of Courcelle's Theorem for treewidth [4].

1. Introduction

In this paper we give an alternate proof of the theorem by Courcelle, Makowski and Rotics [7]: *Every decision or optimization problem expressible in* MSO_1 *is linear time solvable on graphs of bounded cliquewidth.* We prove the same theorem for graphs of bounded rankwidth. Since rankwidth and cliquewidth are equivalent width measures in the sense that a graph has bounded rankwidth iff it has bounded cliquewidth, it does not matter which of these width measures is used to state the theorem [24].

[☆]This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant RO 927/8.

[☆☆]A short version of this paper appeared in the proceedings of TAMC 2011 [22]

Email addresses: `langer@cs.rwth-aachen.de` (Alexander Langer), `rossmani@cs.rwth-aachen.de` (Peter Rossmanith), `sikdar@cs.rwth-aachen.de` (Somnath Sikdar)

The proof by Courcelle et al. [7, 8] makes use of the Feferman-Vaught Theorem [11] adapted to MSO (cf. [16, 15]) and MSO transductions (cf., [5]). Understanding this proof requires a reasonable background in logic and as such this proof is out of reach of many practicing algorithmists. An alternative proof of this theorem has been recently published by Ganian and Hliněný [12] who use an automata-theoretic approach to prove the theorem. Our approach to proving this theorem is game-theoretic, an outline of which follows.

It is known that any graph of rankwidth t can be represented by a t-labeled parse tree [12]. Given any integer q, one can define an equivalence relation on the class of all t-labeled graphs as follows: t-labeled graphs G_1 and G_2 are equivalent, denoted $G_1 \equiv_q^{\text{MSO}} G_2$, iff for every MSO_1-formula φ of quantifier rank at most q, we have: $G_1 \models \varphi$ iff $G_2 \models \varphi$, i.e., no formula with at most q nested quantifiers can distinguish them. Here is a sketch of our proof.

- The number of equivalence classes of the relation $\equiv_q^{\text{MSO}}$ on the class of t-labeled graphs depends only on the quantifier rank q and the number of labels t.

- Each equivalence class can be represented by a tree-like structure of size $f(q,t)$, where f is a computable function of q and t only. This tree-like representative of an equivalence class, called a *reduced characteristic tree of depth* q and denoted by $\text{RC}_q(G)$, captures all model-checking games (defined later) that can be played on graphs G in that equivalence class and formulas of quantifier rank at most q.

- One can construct a reduced characteristic tree of depth q given a t-labeled parse tree of an n-vertex graph in time $O(f'(q,t) \cdot n)$.

- Finally to decide whether $G \models \varphi$, for some MSO_1-formula φ of quantifier rank at most q, we simply simulate the model checking game on φ and G using $\text{RC}_q(G)$. This takes an additional $O(f(q,t))$ time and shows that one can decide whether $G \models \varphi$ in time $O(f''(q,t) \cdot n)$, proving the theorem.

The notions of q-equivalence $\equiv_q^{\text{MSO}}$ and related two-player pebble games (such as the Ehrenfeucht-Fraïssé game) are fundamental to finite model theory and can be found in any book on the subject (cf. [10]). However for understanding this paper, one does not need any prior knowledge of these concepts.

The rest of the paper is organized as follows. Section 2 recaps the basic definitions and properties of rankwidth. Section 3 is a brief introduction to monadic second order logic for those who wish to see it, and has been included to make the paper self-contained. In Section 4, we introduce the equivalence relation $\equiv_q^{\text{MSO}}$, model-checking games and characteristic trees of depth q. In this section we prove that reduced characteristic trees of depth q for t-labeled graphs indeed characterize the equivalence relation $\equiv_q^{\text{MSO}}$ on the class of all t-labeled graphs, and that they have size at most $f(q,t)$, for some computable function of q and t alone. In Section 5 we show how to construct reduced characteristic trees of depth q for an n-vertex graph given its t-labeled parse tree decomposition in time $O(f'(q,t) \cdot n)$. We then use all the ingredients to prove the main theorem. We conclude in Section 6 with a brief discussion of this approach and how it can be used to obtain other results.

2. Rankwidth: Definitions and Basic Properties

Rankwidth is a graph width measure that expresses the structural complexity of graphs. It was introduced by Oum and Seymour to study cliquewidth, another graph width measure [24]. Their main objective was to investigate whether there is an algorithm that takes a graph G and an integer k as input, and decides whether G has cliquewidth at most k in time $O(f(k) \cdot |V(G)|^{O(1)})$. In the parlance of parameterized complexity this means that deciding whether a graph has cliquewidth at most k is fixed-parameter tractable (FPT) w.r.t. k. This question is still open but Oum and Seymour showed that rankwidth and cliquewidth are equivalent width measures in the sense that a graph has bounded rankwidth if and only if it has bounded cliquewidth. They obtained the following relationship between rankwidth and cliquewidth:

$$\text{rankwidth} \leq \text{cliquewidth} \leq 2^{1+\text{rankwidth}} - 1.$$

Moreover they also showed that there does indeed exist an algorithm that decides whether a graph G has rankwidth at most k in time $O(f(k) \cdot |V(G)|^3)$. That is, deciding whether a graph has rankwidth at most k is fixed-parameter tractable w.r.t. k.

We shall briefly recap the basic definitions and properties of rankwidth. The presentation follows [12, 23]. To define rankwidth, it is advantageous to first consider the notion of branchwidth since rankwidth is usually defined in terms of branchwidth.

Branchwidth. Let X be a finite set and let λ be an integer-valued function on the subsets of X. We say that the function λ is *symmetric* if for all $Y \subseteq X$ we have $\lambda(Y) = \lambda(X \setminus Y)$. A *branch-decomposition* of λ is a pair (T, μ), where T is a subcubic tree (a tree with degree at most three) and $\mu\colon X \to \{\, t \mid t \text{ is a leaf of } T\,\}$. For an edge e of T, the connected components of $T \setminus e$ partition the set of leaves of T into disjoint sets L_1 and L_2. The *width* of the edge e of the branch-decomposition (T, μ) is $\lambda(\mu^{-1}(L_1))$. The *width* of (T, μ) is the maximum width over all edges of T. The *branchwidth* of λ is the minimum width of all branch-decompositions of λ.

The branchwidth of a graph G, for instance, is defined by letting $X = E(G)$ and $\lambda(Y)$ to be the number of vertices that are incident to an edge in Y and in $E(G) \setminus Y$ in the above definition.

Rankwidth. Given a graph $G = (V, E)$ and a bipartition (Y_1, Y_2) of the vertex set V, define a binary matrix $A[Y_1, Y_2]$ with rows indexed by the vertices in Y_1 and columns indexed by the vertices in Y_2 as follows: the (u, v)th entry of $A[Y_1, Y_2]$ is 1 if and only if $\{u, v\} \in E$. The *cut-rank* function of G is the function $\rho\colon 2^V \to \mathbf{Z}$ defined as follows: for all $Y \subseteq V$

$$\rho(Y) = \operatorname{rank}(A[Y, V \setminus Y]).$$

The cut-rank function is clearly symmetric. A *rank-decomposition* of G is a branch-decomposition of the cut-rank function on $V(G)$ and the *rankwidth* of G is the branch-width of the cut-rank function.

An important result concerning rankwidth is that there is an FPT-algorithm that constructs a width-k rank-decomposition of a graph G, if there exists one, in time $O(n^3)$ for a fixed value of k.

Theorem 1. [19] *Let k be a constant and $n \geq 2$. Given an n-vertex graph G, one can either construct a rank-decomposition of G of width at most k or confirm that the rankwidth of G is larger than k in time $O(n^3)$.*

2.1. Rankwidth and Parse Tree Decompositions

The definition of rankwidth in terms of branchwidth is the one that was originally proposed by Oum and Seymour in [24]. It is simple and it allows one to prove several properties of rankwidth including the fact that rankwidth and cliquewidth are, in fact, equivalent width measures in the sense that a graph has bounded rankwidth if and only if it has bounded cliquewidth. However this definition is not very useful from an algorithmic point-of-view and this prompted Courcelle and Kanté [6] to introduce an equivalent formulation of rankwidth in terms of certain algebraic operations on labeled graphs. This was restated by Ganian and Hliněný [12] in terms of labeling joins and parse trees which we briefly describe here.

t-labeled graphs. A *t-labeling lab* of a graph G is a mapping $lab\colon V(G) \to 2^{[t]}$ which assigns to each vertex of G a subset of $[t] = \{1, \ldots, t\}$. A *t-labeled graph* is a pair (G, lab), where lab is a labeling of G and is denoted by $\bar{G}$. Since a t-labeling function may assign the empty label to each vertex, an unlabeled graph is considered to be a t-labeled graph for all $t \geq 1$. A t-labeling of G may also be interpreted as a mapping from $V(G)$ to the t-dimensional binary vector space $\mathrm{GF}(2^t)$ by associating the subset $X \subseteq [t]$ with the t-bit vector $\mathbf{x} = x_1 \ldots x_t$, where $x_i = 1$ if and only if $i \in X$. Thus one can represent a t-labeling lab of an n-vertex graph as an $n \times t$ binary matrix. This interpretation will prove useful later on when t-joins are discussed.

A *t-relabeling* is a linear transformation from the space $\mathrm{GF}(2^t)$ to $\mathrm{GF}(2^t)$ and one can therefore represent a t-relabeling by a $t \times t$ binary matrix T_f. We represent a t-relabeling f as a function $f\colon 2^{[t]} \to 2^{[t]}$. For a t-labeled graph $\bar{G} = (G, lab)$, we define $f(\bar{G})$ to be the t-labeled graph $(G, f \circ lab)$, where $(f \circ lab)(v)$ is the vector in $\mathrm{GF}(2^t)$ obtained by applying the linear transformation f to the vector $lab(v)$. It is easy to see that the labeling $lab' = f \circ lab$ is the matrix product $lab \times T_f$.

We now define three operators on t-labeled graphs that will be used to define parse tree decompositions of t-labeled graphs. These operators were first described by Ganian and Hliněný in [12]. The first operator is denoted $\odot$ and represents a nullary operator that creates a new graph vertex with the label 1. The second operator is the t-labeled join and is defined as follows. Let $\bar{G}_1 = (G_1, lab_1)$ and $\bar{G}_2 = (G_2, lab_2)$ be t-labeled graphs. The *t-labeled join* of $\bar{G}_1$ and $\bar{G}_2$, denoted $\bar{G}_1 \otimes \bar{G}_2$, is defined as taking the disjoint union of G_1 and G_2 and adding all edges between vertices $u \in V(G_1)$ and $v \in V(G_2)$ such that $|lab_1(u) \cap lab_2(v)|$ is odd. The resulting graph is unlabeled.

Note that $|lab_1(u) \cap lab_2(v)|$ is odd if and only if the scalar product $lab_1(u) \bullet lab_2(v) = 1$, that is, the vectors $lab_1(u)$ and $lab_2(v)$ are *not orthogonal* in the space $\mathrm{GF}(2^t)$. For $X \subseteq V(G_1)$, the set of vectors $\gamma(\bar{G}_1, X) = \{\, lab_1(u) \mid u \in X \,\}$ generates a subspace $\langle \gamma(\bar{G}_1, X) \rangle$ of $\mathrm{GF}(2^t)$. The following result shows which pairs of vertex subsets do not generate edges in a t-labeled join operation.

Proposition 1. [13] *Let $X \subseteq V(G_1)$ and $Y \subseteq V(G_2)$ be arbitrary nonempty subsets of t-labeled graphs $\bar{G}_1$ and $\bar{G}_2$. In the join graph $\bar{G}_1 \otimes \bar{G}_2$ there is no edge between any vertex of X and a vertex of Y if and only if the subspaces $\langle \gamma(\bar{G}_1, X) \rangle$ and $\langle \gamma(\bar{G}_2, Y) \rangle$ are orthogonal in the vector space* $\mathrm{GF}(2^t)$.

The third operator is called the t-labeled composition and is defined using the t-labeled join and t-relabelings. Given three t-relabelings $g, f_1, f_2\colon 2^{[t]} \to 2^{[t]}$, the *$t$-labeled composition* $\otimes[g|f_1, f_2]$ is defined on a pair of t-labeled graphs $\bar{G}_1 = (G_1, lab_1)$ and $\bar{G}_2 = (G_2, lab_2)$ as follows:

$$\bar{G}_1 \otimes[g|f_1, f_2]\, \bar{G}_2 := \bar{H} = (\bar{G}_1 \otimes g(\bar{G}_2), lab),$$

where $lab(v) = f_i \circ lab_i(v)$ for $v \in V(G_i)$ and $i \in \{1, 2\}$. Thus the t-labeled composition first performs a t-labeling join of $\bar{G}_1$ and $g(\bar{G}_2)$ and then relabels the vertices of G_1 using f_1 and the vertices of G_2 with f_2. Note that a t-labeling composition is not commutative and that $\{u, v\}$ is an edge of $\bar{H}$ if and only if $lab_1(u) \bullet (lab_2(v) \times T_g) = 1$, where T_g is the matrix representing the linear transformation g.

Definition 1 (t-labeled Parse Trees)**.** A *t-labeled parse tree* T is a finite, ordered, rooted subcubic tree (with the root of degree at most two) such that

1. all leaves of T are labeled with the $\odot$ symbol, and
2. all internal nodes of T are labeled with a t-labeled composition symbol.

A parse tree T *generates* the graph G that is obtained by the successive leaves-to-root application of the operators that label the nodes of T.

The next result shows that rankwidth can be defined using t-labeled parse trees.

Theorem 2 (Rankwidth Parsing Theorem [6, 12])**.** *A graph G has rankwidth at most t if and only if some labeling of G can be generated by a t-labeled parse tree. Moreover, a width-t rank-decomposition of an n-vertex graph can be transformed into a t-labeled parse tree on $\Theta(n)$ nodes in time $O(t^2 \cdot n^2)$.*

We now proceed to show the following.

The Main Theorem. [7, 12] *Let φ be an* MSO_1*-formula with* $\text{qr}(\varphi) \leq q$. *There is an algorithm that takes as input a t-labeled parse tree decomposition T of a graph G and decides whether $G \models \varphi$ in time $O(f(q, t) \cdot |T|)$, where f is some computable function and $|T|$ is the number of nodes in T.*

Here is how the sequel is organized. In Section 3 we briefly introduce monadic second order logic. In Section 4 we introduce a construct that plays a key role in our proof of the Main Theorem. This construct, called a characteristic tree of depth q, is important for three reasons. Firstly, a characteristic tree of depth q for a graph G allows one to test whether an MSO formula φ of quantifier rank at most q holds in G. Secondly, a characteristic tree has small size and, thirdly, it can be efficiently constructed for graphs of bounded rankwidth. The construction of characteristic trees is described in Section 5, where we also prove the main theorem.

3. An Introduction to MSO Logic

In this section, we present a brief introduction to monadic second order logic. We follow Ebbinghaus and Flum [10]. *Monadic second-order logic (MSOL)* is an extension of first-order logic which allows quantification over sets of objects. To define the syntax of MSO, fix a *vocabulary* τ which is a finite set of relation symbols $P, Q, R, \ldots$ each associated with a natural number known as its *arity*.

A *structure* $\mathscr{A}$ *over vocabulary* τ (also called a τ*-structure*) consists of a set A called the *universe* of $\mathscr{A}$ and a p-ary relation $R^{\mathscr{A}} \subseteq A \times \cdots \times A$ (p times) for every p-ary relation symbol R in τ. If the universe is empty then we say that the structure is *empty*. Graphs can be expressed in a natural way as relational structures with universe the vertex set and a vocabulary consisting of a single binary (edge) relation symbol. To express a t-labeled graph G, we may use a vocabulary τ consisting of the binary relation symbol E (representing, as usual, the edge relation) and t unary relation symbols $L_1, \ldots, L_t$, where L_i represents the set of vertices labeled i.

A formula in MSO is a string of symbols from an alphabet that consists of

- the *relation symbols* of τ
- a countably infinite set of *individual variables* $x_1, x_2, \ldots$
- a countably infinite set of *set variables* $X_1, X_2, \ldots$
- $\neg$, $\vee$, $\wedge$ (the connectives *not*, *or*, *and*)
- $\exists$, $\forall$ (the *existential quantifier* and the *universal quantifier*)
- $=$ (the *equality* symbol)
- (,) (the *bracket* symbols).

The *formulas* of MSO over the vocabulary τ are strings that are obtained from finitely many applications of the following rules:

1. If t_1 and t_2 are individual (respectively, set) variables then $t_1 = t_2$ is a formula.
2. If R is an p-ary relation symbol in τ and $t_1, \ldots, t_r$ are individual variables, then $Rt_1, \ldots, t_r$ is a formula.
3. If X is a set variable and t is an individual variable then Xt is a formula.
4. If φ is a formula then $\neg\varphi$ is a formula.
5. If φ and ψ are formulas then $(\varphi \vee \psi)$ is a formula.
6. If φ and ψ are formulas then $(\varphi \wedge \psi)$ is a formula.
7. If φ is a formula and x an individual variable then $\exists x\varphi$ is a formula.
8. If φ is a formula and x an individual variable then $\forall x\varphi$ is a formula.
9. If φ is a formula and X a set variable then $\exists X\varphi$ is a formula.
10. If φ is a formula and X a set variable then $\forall X\varphi$ is a formula.

The formulas obtained by 1, 2, or 3 above are *atomic formulas*. Formulas of types 6, 8, and 10 are called *universal*, and formulas of types 5, 7, and 9 are *existential*.

The *quantifier rank* $\mathrm{qr}(\varphi)$ of a formula φ is the maximum number of nested quantifiers occurring in it.

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{qr}(\varphi) & := & 0, \text{ if } \varphi \text{ is atomic}; & \mathrm{qr}(\exists x\varphi) & := & \mathrm{qr}(\varphi)+1; \\
\mathrm{qr}(\neg\varphi) & := & \mathrm{qr}(\varphi); & \mathrm{qr}(\exists X\varphi) & := & \mathrm{qr}(\varphi)+1; \\
\mathrm{qr}(\varphi \vee \psi) & := & \max\{\mathrm{qr}(\varphi), \mathrm{qr}(\psi)\}; & \mathrm{qr}(\forall x\varphi) & := & \mathrm{qr}(\varphi)+1. \\
\mathrm{qr}(\forall X\varphi) & := & \mathrm{qr}(\varphi)+1; & & &
\end{array}$$

A variable in a formula is *free* if it is not within the scope of a quantifier. A formula without free variables is called a *sentence*. By $\mathrm{free}(\varphi)$ we denote the set of free variables of φ.

We now assign meanings to the logical symbols by defining the *satisfaction relation* $\mathscr{A} \models \varphi$. Let $\mathscr{A}$ be a τ-structure. An *assignment* in $\mathscr{A}$ is a function α that assigns individual variables values in A and set variables subsets of A.

For an individual variable x and an assignment α, we let $\alpha[x/a]$ denote an assignment that agrees with α except that it assigns the value $a \in A$ to x. The symbol $\alpha[X/B]$ has the same meaning for a set variable X and a set $B \subseteq A$. We define the relation $\mathscr{A} \models \varphi[\alpha]$ (φ is true in $\mathscr{A}$ under α) as follows:

$$\begin{array}{lll}
\mathscr{A} \models t_1 = t_2[\alpha] & \text{iff} & \alpha(t_1) = \alpha(t_2) \\
\mathscr{A} \models Rt_1 \ldots t_n[\alpha] & \text{iff} & R^{\mathscr{A}} \alpha(t_1) \ldots \alpha(t_n) \\
\mathscr{A} \models \neg\varphi[\alpha] & \text{iff} & \text{not } \mathscr{A} \models \varphi[\alpha] \\
\mathscr{A} \models (\varphi \vee \psi)[\alpha] & \text{iff} & \mathscr{A} \models \varphi[\alpha] \text{ or } \mathscr{A} \models \psi[\alpha] \\
\mathscr{A} \models (\varphi \wedge \psi)[\alpha] & \text{iff} & \mathscr{A} \models \varphi[\alpha] \text{ and } \mathscr{A} \models \psi[\alpha] \\
\mathscr{A} \models \exists x\varphi[\alpha] & \text{iff} & \text{there is an } a \in A \text{ such that } \mathscr{A} \models \varphi[\alpha[x/a]] \\
\mathscr{A} \models \forall x\varphi[\alpha] & \text{iff} & \text{for all } a \in A \text{ it holds that } \mathscr{A} \models \varphi[\alpha[x/a]] \\
\mathscr{A} \models \exists X\varphi[\alpha] & \text{iff} & \text{there exists } B \subseteq A \text{ such that } \mathscr{A} \models \varphi[\alpha[X/B]] \\
\mathscr{A} \models \forall X\varphi[\alpha] & \text{iff} & \text{for all } B \subseteq A \text{ it holds that } \mathscr{A} \models \varphi[\alpha[X/B]]
\end{array}$$

4. The $\equiv_q^{\mathrm{MSO}}$-Relation and its Characterization

Given a vocabulary τ and a natural number q, one can define an equivalence relation on the class of τ-structures as follows. For τ-structures $\mathscr{A}$ and $\mathscr{B}$ and $q \in \mathbf{N}$, define $\mathscr{A} \equiv_q^{\mathrm{MSO}} \mathscr{B}$ (q-equivalence) if and only if $\mathscr{A} \models \varphi \iff \mathscr{B} \models \varphi$ for all MSO sentences φ of quantifier rank at most q. In other words, two structures are q-equivalent if and only if no sentence of quantifier rank at most q can distinguish them.

We provide a characterization of the relation $\equiv_q^{\mathrm{MSO}}$ using objects called characteristic trees of depth q. We show that two τ-structures $\mathscr{A}$ and $\mathscr{B}$ have identical characteristic trees of depth q if and only if $\mathscr{A} \equiv_q^{\mathrm{MSO}} \mathscr{B}$. We shall see that characteristic trees are specially useful because their size is "small" and for graphs of bounded rankwidth can be constructed efficiently given their parse tree decomposition. However before we can do that, we need a few definitions.

Definition 2 (Induced Structure and Sequence)**.** Let $\mathscr{A}$ a τ-structure with universe A and let $\bar{c} = c_1, \ldots, c_m \in A^m$. The *structure* $\mathscr{A}' = \mathscr{A}[\bar{c}] = \mathscr{A}[\{c_1, \ldots, c_m\}]$

induced by $\bar{c}$ is a τ-structure with universe $A' = \{c_1, \ldots, c_m\}$ and interpretations $P^{\mathscr{A}'} := P^{\mathscr{A}} \cap \{c_1, \ldots, c_m\}^r$ for every relation symbol $P \in \tau$ of arity r. For a set $U \subseteq A$, we let $\bar{c}[U]$ be the subsequence of $\bar{c}$ that contains only objects in U.

Definition 3 (Intersection, Union and Concatenation of Sequences)**.** *Let A be a set and $U \subseteq A$; let $\bar{c} = c_1, \ldots, c_m \in A^m$, $\bar{C} = C_1, \ldots, C_p, \bar{D} = D_1, \ldots, D_p$ where $C_i, D_i \subseteq A$. We let $\bar{C} \cap U$, $\bar{C} \cap \bar{c}$ and $\bar{C} \cap \bar{D}$ to denote (respectively) the sequences $C_1 \cap U, \ldots, C_p \cap U$, $C_1 \cap \{c_1, \ldots, c_m\}, \ldots, C_p \cap \{c_1, \ldots, c_m\}$ and $C_1 \cap D_1, \ldots, C_p \cap D_p$. We let $\bar{C} \cup \bar{D}$ to denote $C_1 \cup D_1, \ldots, C_p \cup D_p$. For $a \in A$, we let $\bar{c} \cdot a$ the concatenation of the sequence $\bar{c}$ with a. We usually omit the $\cdot$ while writing concatenations.*

Therefore when it comes to the union and intersection of sequences, we always mean their componentwise union or intersection.

Definition 4 (Partial Isomorphism)**.** Let $\mathscr{A}$ and $\mathscr{B}$ be structures over the vocabulary τ with universes A and B, respectively, and let π be a map such that domain$(\pi) \subseteq A$ and range$(\pi) \subseteq B$. The map π is said to be a *partial isomorphism* from $\mathscr{A}$ to $\mathscr{B}$ if

1. π is one-to-one and onto;
2. for every p-ary relation symbol $R \in \tau$ and all $a_1, \ldots, a_p \in$ domain(π),

$$R^{\mathscr{A}} a_1, \ldots, a_p \qquad \text{iff} \qquad R^{\mathscr{B}} \pi(a_1), \ldots, \pi(a_p).$$

If domain$(\pi) = A$ and range$(\pi) = B$, then π is an *isomorphism* between $\mathscr{A}$ and $\mathscr{B}$ and $\mathscr{A}$ and $\mathscr{B}$ are *isomorphic.*

Let $(\mathscr{A}, \bar{C})$ and $(\mathscr{B}, \bar{D})$ be tuples, where $\bar{C} = A_1, \ldots, A_s$ and $\bar{D} = B_1, \ldots, B_s$, $s \geq 0$, such that for all $1 \leq i \leq s$, we have $A_i \subseteq A$ and $B_i \subseteq B$. We say that π is a partial isomorphism between $(\mathscr{A}, \bar{C})$ and $(\mathscr{B}, \bar{D})$ if

1. π is a partial isomorphism between $\mathscr{A}$ and $\mathscr{B}$,
2. for each $a \in$ domain(π) and all $1 \leq i \leq s$, it holds that $a \in A_i$ iff $\pi(a) \in B_i$.

The tuples $(\mathscr{A}, \bar{C})$ and $(\mathscr{B}, \bar{D})$ are *isomorphic* if π is an isomorphism between $\mathscr{A}$ and $\mathscr{B}$ and, in addition, condition (2) above holds.

In Definition 2 of an induced structure we ignore the order of the elements in $\bar{c}$. For the purposes in this paper, the order in which the elements are chosen is important because it is used to map variables in the formula to elements in the structure. Moreover, elements could repeat in the vector $\bar{c}$ and this fact is lost when we consider the induced structure $\mathscr{A}[\bar{c}]$. To capture both the order and the multiplicity of the elements in vector $\bar{c}$ in the structure $\mathscr{A}[\bar{c}]$, we introduce the notion of an *ordered induced structure*.

Let U be a set and $\equiv$ be an equivalence relation on U. For $u \in U$, we let $[u]_\equiv = \{\, u' \in U \mid u \equiv u' \,\}$ be the *equivalence class* of u under $\equiv$, and $U/{\equiv} = \{\, [u]_\equiv \mid u \in U\}$ be the *quotient space* of U under $\equiv$.

A vector $\bar{c} = c_1, \ldots, c_m \in A^m$ defines a natural equivalence relation $\equiv_{\bar{c}}$ on the set $[m] = \{1, \ldots, m\}$: for $i, j \in [m]$, we have $i \equiv_{\bar{c}} j$ if and only if $c_i = c_j$. For simplicity, we shall write $[i]_{\bar{c}}$ for $[i]_{\equiv_{\bar{c}}}$.

Definition 5 (Ordered Induced Structure)**.** Let $\mathscr{A}$ be a τ-structure with universe A and $\bar{c} = c_1, \ldots, c_m \in A^m$. The *ordered structure induced by* $\bar{c}$ is the τ-structure $\mathscr{H} = \mathrm{Ord}(\mathscr{A}, \bar{c})$ with universe $H = [m]/{\equiv_{\bar{c}}}$ such that the map $h\colon c_i \mapsto [i]_{\bar{c}}$, $1 \le i \le m$, is an isomorphism between $\mathscr{A}[\bar{c}]$ and $\mathscr{H}$.

Let $\bar{C} = C_1, \ldots, C_p$ with $C_i \subseteq A$, $1 \le i \le p$. Then we let

$$\mathrm{Ord}(\mathscr{A}, \bar{c}, \bar{C}) := \left(\mathrm{Ord}(\mathscr{A}, \bar{c}), \bar{h}, h(\bar{C} \cap \bar{c})\right),$$

where $h\colon c_i \mapsto [i]_{\bar{c}}$, $1 \le i \le m$, $\bar{h} = h(c_1), \ldots, h(c_m)$ and $h(\bar{C} \cap \bar{c}) = h(C_1 \cap \bar{c}), \ldots, h(C_p \cap \bar{c})$.

Thus an ordered structure $\mathscr{H} = \mathrm{Ord}(\mathscr{A}, \bar{c})$ induced by $\bar{c}$ is simply the structure $\mathscr{A}[\bar{c}]$ with element c_i being called $[i]_{\bar{c}}$. See Figure 1 for an example.

4.1. Model Checking Games and Characteristic Trees

Testing whether a non-empty structure models a formula can be specified by a *model checking game* (also known as *Hintikka game*, see [18, 14]). Let $\mathscr{A}$ be a

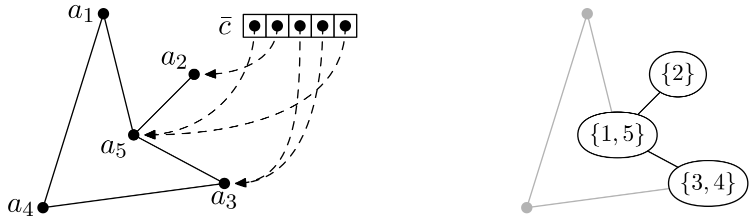

Figure 1: The vector $\bar{c} = a_5a_2a_3a_3a_5$ lists vertices in the graph $\mathscr{G}$ on the left. The resulting ordered induced structure $\mathrm{Ord}(\mathscr{G}, \bar{c})$ is depicted in black on the right. Note that essentially each vertex in $\mathscr{G}[\bar{c}]$ is renamed to the set of positions in which it appears in the vector $\bar{c}$.

τ-structure with universe A. Let φ be a formula and α be an assignment to the free variables of φ. The game is played between two players called the *verifier* and the *falsifier*. The verifier tries to prove that $\mathscr{A} \models \varphi[\alpha]$ whereas the falsifier tries to disprove this. We assume without loss of generality that φ is in negation normal form, i.e., negations in φ appear only at the atomic level. This can always be achieved by applying simple rewriting rules such as $\neg\forall x\varphi(x) \rightsquigarrow \exists x\neg\varphi(x)$.

The model checking game $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ is positional with positions (ψ, β), where ψ is a subformula of φ and β is an assignment to the free variables of ψ. The game starts at position (φ, α). At a position $(\forall X\psi(X), \beta)$, the falsifier chooses a subset $D \subseteq A$, and the game continues at position $(\psi, \beta[X/D])$. Similarly, at a position $(\forall x\psi(x), \beta)$ or $(\psi_1 \wedge \psi_2, \beta)$, the falsifier chooses an element $d \in A$ or some $\psi := \psi_i$ for some $1 \leq i \leq 2$ and the game then continues at position $(\psi, \beta[x/d])$ or (ψ, β), respectively. The verifier moves analogously at existential formulas. Note that since the structure of the formula determines which player gets to make a move, it might well be that a player has to make several moves before the second has the right to make a move. If an element is chosen then the move is called a *point move*; if a set is chosen then the move is a *set move*. The game ends once a position (ψ, β) is reached, such that ψ is an atomic or negated formula. The verifier *wins* if and only if $\mathscr{A} \models \psi[\beta]$. We say that the verifier has a *winning strategy* if they win every play of the game irrespective of the choices made by the falsifier. In what follows, we identify a position (ψ, β) of the game $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ with the game $\mathcal{MC}(\mathscr{A}, \psi, \beta)$.

It is well known that the model checking game characterizes the satisfaction relation $\models$. The following lemma can easily be shown by induction over the structure of φ.

Lemma 1 (cf., [14])**.** *Let $\mathscr{A}$ be a τ-structure, let φ be an* MSO *formula, and let α be an assignment to the free variables of φ. Then $\mathscr{A} \models \varphi[\alpha]$ if and only if the verifier has a winning strategy on the model checking game on $\mathscr{A}$, φ, and α.*

A model checking game on a τ-structure $\mathscr{A}$ and a formula φ with quantifier rank q can be represented by a tree of depth q in which the nodes represent positions in the game and the edges represent point and set moves made by the players. Such a tree is called a *game tree* and is used in combinatorial game theory for analyzing games (see [2], for instance).

For our purposes, we define a notion related to game trees called *full characteristic trees* which are finite rooted trees, where the nodes represent positions and edges represent moves of the game. A node is a tuple that represents the sets and elements that have been chosen thus far. The node can be thought of as a succinct representation of the state of the game played till the position represented by that node. However, note that a full characteristic tree depends on the quantifier rank q and *not* on a particular formula.

Definition 6 (Full Characteristic Trees)**.** Let $\mathscr{A}$ be a τ-structure with universe A and let $q \in \mathbf{N}$. For elements $\bar{c} = c_1, \dots, c_m \in A^m$, sets $\bar{C} = C_1, \dots, C_p$ with $C_i \subseteq A$, $1 \le i \le p$, let $T = \mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$ be a finite rooted tree such that

1. $\mathrm{root}(T) = (\mathscr{A}[\bar{c}], \bar{c}, \bar{C} \cap \bar{c})$,
2. if $m + p + 1 \le q$ then the subtrees of the root of $\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$ is the set

$$\left\{ \mathrm{FC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \;\middle|\; d \in A \right\} \cup \left\{ \mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}D) \;\middle|\; D \subseteq A \right\}.$$

The *full characteristic tree of depth q* for $\mathscr{A}$, denoted by $\mathrm{FC}_q(\mathscr{A})$, is defined as $\mathrm{FC}_q(\mathscr{A}, \varepsilon, \varepsilon)$, where ε is the empty sequence.

Let $T = (V, E)$ be a rooted tree. We let $\mathrm{root}(T)$ be the root of T and for $u \in V$ we let

$$\mathrm{children}_T(u) = \{ v \in V \mid (u, v) \in E \text{ and } \mathrm{dist}_T(\mathrm{root}(T), u) < \mathrm{dist}_T(\mathrm{root}(T), v) \},$$

where $\mathrm{dist}(x, y)$ denotes the length of the shortest path between x and y. We also let $\mathrm{subtree}_T(u)$ be a subtree of T rooted at u, and $\mathrm{subtrees}(T) = \{\, \mathrm{subtree}_T(u) \mid u \in \mathrm{children}_T(\mathrm{root}(T)) \,\}$.

We now define a model checking game $\mathcal{MC}(F, \varphi, \bar{x}, \bar{X})$ on full characteristic trees $F = \mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$ and formulas φ with $\mathrm{qr}(\varphi) \leq q$, where $\bar{x} = x_1, \ldots, x_m$ are the free object variables of φ, $\bar{X} = X_1, \ldots, X_p$ are the free set variables of φ, $\bar{c} = c_1, \ldots, c_m \in A^m$, and $\bar{C} = C_1, \ldots, C_p$ with $C_i \subseteq A$, $1 \leq i \leq p$. The rules are similar to the classical model checking game $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$. The game is positional and played by two players called the *verifier* and the *falsifier* and is defined over subformulas ψ of φ. However instead of choosing sets and elements explicitly, the tree F is traversed top-down. At the same time, we "collect" the variables the players encountered, such that we can make the assignment explicit once the game ends. The game starts at the position $(\varphi, \bar{x}, \bar{X}, \mathrm{root}(F))$. Let $(\psi, \bar{y}, \bar{Y}, v)$ be the position at which the game is being played, where $v = (\mathscr{H}, \bar{d}, \bar{D})$ is a node of $\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$, and ψ is a subformula of φ with $\mathrm{free}(\psi) = \bar{y} \cup \bar{Y}$. At a position $(\forall X \vartheta(X), \bar{y}, \bar{Y}, v)$ the falsifier chooses a child $u = (\mathscr{H}, \bar{d}, \bar{D}D)$ of v, where $D \subseteq A$, and the game continues at position $(\vartheta, \bar{y}, \bar{Y}X, u)$. Similarly, at a position $(\forall x \vartheta(x), \bar{y}, \bar{Y}, v)$ the falsifier chooses a child $u = (\mathscr{H}', \bar{d}d, \bar{D})$, where $d \in A$, and the game continues in $(\vartheta, \bar{y}x, \bar{Y}, u)$, and at a position $(\vartheta_1 \wedge \vartheta_2, \bar{y}, \bar{Y}, v)$, the falsifier chooses some $1 \leq i \leq 2$, and the game continues at position $(\vartheta_i, \bar{y}, \bar{Y}, v)$. The verifier moves analogously at existential formulas.

The game stops once an atomic or negated formula has been reached. Suppose that a particular play of the game ends at a position $(\psi, \bar{y}, \bar{Y}, v)$, where ψ is a negated atomic or atomic formula with

$$\mathrm{free}(\psi) = \{y_1, \ldots, y_s, Y_1, \ldots, Y_t\}$$

and $v = (\mathscr{H}, \bar{d}, \bar{D})$ some node of F, where $\bar{d} = d_1, \ldots, d_s$ and $\bar{D} = D_1, \ldots, D_t$. Let α be an assignment to the free variables of φ, such that $\alpha(y_i) = d_i$, $1 \leq i \leq s$, and $\alpha(Y_i) = D_i$, $1 \leq i \leq t$. The verifier *wins* the game if and only if $\mathscr{H} \models \psi[\alpha]$. The verifier has a *winning strategy* if and only if they can win every play of the game irrespective of the choices made by the falsifier. In what follows, we identify a position $(\psi, \bar{y}, \bar{Y}, v)$ of the game $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$, where $v = (\mathscr{H}, \bar{d}, \bar{D})$, with the game $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{d}, \bar{D}), \psi, \bar{y}, \bar{Y})$.

Lemma 2. *Let $\mathscr{A}$ be a τ-structure and let φ be an* MSO *formula with* $\mathrm{qr}(\varphi) \leq q$ *and free variables $\{x_1, \ldots, x_m, X_1, \ldots, X_m\}$. Let α be an assignment to the free variables of φ. Then the verifier has a winning strategy in the model checking game $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ if and only if the verifier has a winning strategy in the model checking game* $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$*, where $\bar{c} = \alpha(x_1), \ldots, \alpha(x_m)$ and $\bar{C} = \alpha(X_1), \ldots, \alpha(X_p)$.*

Proof. The proof consists in observing that any play of the model checking game $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ can be simulated in $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$ and vice versa.

For assume that $\mathrm{qr}(\varphi) = q$ (otherwise, pad φ with quantifiers). The proof is by an induction on $q - m - p$ and the structure of φ. If $q = 0$ and φ is an atomic or negated atomic formula, then the verifier wins $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ if and only if $\mathscr{A} \models \varphi[\alpha]$ if and only if $\mathscr{A}[\bar{c}] \models \varphi[\alpha]$, where $\mathrm{root}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})) = (A[\bar{c}], \bar{c}, \bar{C} \cap \bar{c})$, and hence if and only if the verifier wins $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$.

If $q > 0$ and $\varphi = \forall x \psi(x)$, then the verifier has a winning strategy for $\mathcal{MC}(\mathscr{A}, \varphi, \alpha)$ if and only if they have a winning strategy for $\mathcal{MC}(\mathscr{A}, \psi, \alpha[x/a])$ for all $a \in A$. For each such $a \in A$, by the induction hypothesis the verifier has a winning strategy in $\mathcal{MC}(\mathscr{A}, \psi, \alpha[x/a])$ if and only they have a winning strategy in the model checking game $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}a, \bar{C}), \psi, \bar{x}x, \bar{X})$. At position $(\exists x \psi(x), \bar{x}, \bar{X}, v)$, where $v = \mathrm{root}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}))$, the falsifier chooses a child $u = (\mathscr{H}, \bar{c}a, \bar{C})$ of v, where $a \in A$, and the game continues at position $(\psi, \bar{x}x, \bar{X}, u)$. Hence, the verifier has a winning strategy in $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$ if and only if they have a winning strategy on $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}a, \bar{C}), \varphi, \bar{x}x, \bar{X})$, and the claim follows.

The remaining cases follow analogously.

□

Lemma 2 showed that a full characteristic tree of depth q for a structure $\mathscr{A}$

can be used to simulate the model checking game on $\mathscr{A}$ and any formula φ of quantifier rank at most q. However the size of such a tree is of the order $(2^n+n)^q$, where n is the number of elements in the universe of $\mathscr{A}$. We now show that one can "collapse" equivalent branches of a full characteristic tree to obtain a much smaller labeled tree (called a reduced characteristic tree) that is in some sense equivalent to the original (full) tree. We will then show that for a graph G of rankwidth at most t, the reduced characteristic tree of G is efficiently computable given a t-labeled parse tree decomposition of G. We achieve this collapse by replacing the induced structures $\mathscr{A}[\bar{c}]$ in the full characteristic tree by a more generic, implicit representation — that of their ordered induced substructures $\mathrm{Ord}(\mathscr{A}, \bar{c})$.

Definition 7 (Reduced Characteristic Trees)**.** Let $\mathscr{A}$ be a τ-structure and let $q \in \mathbf{N}$. For elements $\bar{c} = c_1, \ldots, c_m \in A^m$ and sets $\bar{C} = C_1, \ldots, C_p$ with $C_i \subseteq A$, $1 \leq i \leq p$, we let $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ be a finite rooted tree such that

1. $\mathrm{root}(\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})) = \mathrm{Ord}(\mathscr{A}, \bar{c}, \bar{C})$,
2. if $m + p + 1 \leq q$ then the subtrees of the root of $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ is the set

$$\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \mid d \in A \,\} \cup \{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C}D) \mid D \subseteq A \,\}.$$

The *reduced characteristic tree of depth* q for the structure $\mathscr{A}$, denoted by $\mathrm{RC}_q(\mathscr{A})$, is defined to be $\mathrm{RC}_q(\mathscr{A}, \varepsilon, \varepsilon)$, where ε is the empty sequence.

See Figure 2 for an example. One can define the model checking game $\mathcal{MC}(R, \varphi, \bar{x}, \bar{X})$ on a tree $R = \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ in exactly the same manner as $\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}), \varphi, \bar{x}, \bar{X})$. As mentioned before, our interest in $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ lies in that:

1. they are equivalent to $\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$,
2. they are "small"; and,
3. they are efficiently computable if $\mathscr{A}$ is a graph of rankwidth at most t and such a rank decomposition is provided.

We first show that the reduced characteristic tree $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ is equivalent to its full counterpart $\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$.

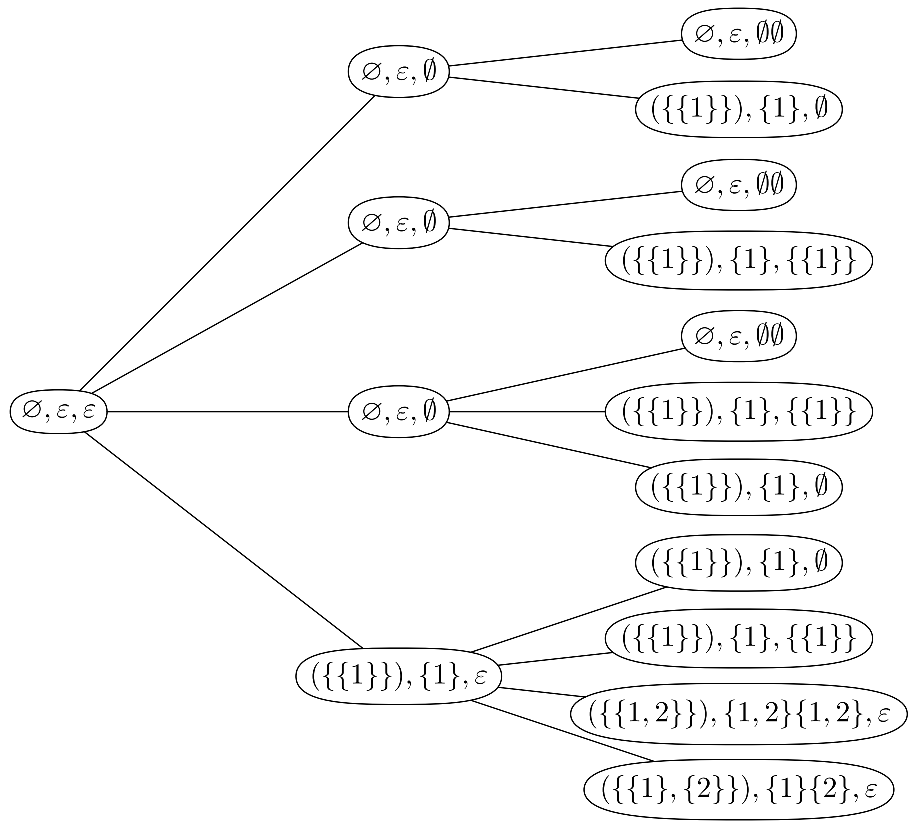

Figure 2: The tree $\mathrm{RC}_2(\mathscr{A})$ for a τ-structure $\mathscr{A}$ with $\tau = \emptyset$ and $A = \{a_1, a_2\}$. Here, $\varnothing$ denotes an empty structure, and $\emptyset\emptyset$ is the sequence of two empty sets. The bottom right node $(\mathscr{H}, \bar{c}, \bar{C}) = \big((\{\{1\}, \{2\}\}), \{1\}\{2\}, \varepsilon\big)$ represents, at the same time, the identical subtrees $\mathrm{RC}_2(\mathscr{A}, a_1a_2, \varepsilon)$ and $\mathrm{RC}_2(\mathscr{A}, a_2a_1, \varepsilon)$. The universe of $\mathscr{H}$ is $H = \{[1]_{a_1a_2}, [2]_{a_1a_2}\} = \{[1]_{a_2a_1}, [2]_{a_2a_1}\} = \{\{1\}, \{2\}\}$, since elements a_1, a_2 and a_2, a_1, respectively, have been chosen in this order. No set has been chosen, hence the empty sequence $\bar{C} = \varepsilon$. Similarly, the next node in that column, $\big((\{\{1, 2\}\}), \{1, 2\}\{1, 2\}, \varepsilon\big)$, represents the trees $\mathrm{RC}_2(\mathscr{A}, a_1a_1, \varepsilon)$ and $\mathrm{RC}_2(\mathscr{A}, a_2a_2, \varepsilon)$. Here the universe is $\{\{1, 2\}\}$ since the same element has been chosen twice. Note that the root node has only four subtrees in total, since $\mathrm{RC}_2(\mathscr{A}, \varepsilon, \{a_1\}) = \mathrm{RC}_2(\mathscr{A}, \varepsilon, \{a_2\})$ (third subtree from the top), and $\mathrm{RC}_2(\mathscr{A}, a_1, \varepsilon) = \mathrm{RC}_2(\mathscr{A}, a_2, \varepsilon)$ (bottom subtree).

Lemma 3. *Let $\mathscr{A}$ be a τ-structure and let $q \in \mathbf{N}$. Let $\bar{c} = c_1, \dots, c_m \in A^m$ and $\bar{C} = C_1, \dots, C_p$ with $C_i \subseteq A$, $1 \le i \le p$. Let $F = \mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$ and $R = \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$. Then the verifier has a winning strategy in the model checking game $\mathcal{MC}(F, \varphi, \bar{x}, \bar{X})$ if and only if the verifier has a winning strategy in the game $\mathcal{MC}(R, \varphi, \bar{x}, \bar{X})$, where $\varphi \in \mathrm{MSO}(\tau)$ with $\mathrm{qr}(\varphi) \le q$ with free object variables $\bar{x} = x_1, \dots, x_m$ and free set variables $\bar{X} = X_1, \dots, X_p$.*

Proof. Without loss of generality, we assume $\mathrm{qr}(\varphi) = q$ (otherwise, pad φ with quantifiers). The proof is by an induction on $q - m - p$ and the structure of φ. If $q = 0$, then

$$\begin{aligned}\mathrm{root}(F) = (\mathscr{A}[\bar{c}], \bar{c}, \bar{C} \cap \bar{c}) \cong \\ (\mathscr{H}, h(c_1) \dots h(c_m), h(\bar{C} \cap \bar{c})) = \mathrm{Ord}(\mathscr{A}, \bar{c}, \bar{C}) = \mathrm{root}(R),\end{aligned}$$

where $h \colon c_i \mapsto [i]_{\bar{c}}$, $1 \le i \le m$ is an isomorphism between $(\mathscr{A}[\bar{c}], \bar{C} \cap \bar{c})$ and $(\mathscr{H}, h(\bar{C} \cap \bar{c}))$. The lemma therefore holds since MSO formulas cannot distinguish isomorphic structures.

Therefore assume that $q > 0$. If $\varphi = (\psi_1 \wedge \psi_2)$ or $\varphi = (\psi_1 \vee \psi_2)$, then the claim immediately follows by the induction hypothesis for ψ_i, $1 \le i \le 2$. Assume therefore that $\varphi = \exists X \psi(X)$ and suppose that the verifier has a winning strategy in one of the games, say, in $\mathcal{MC}(R, \varphi, \bar{x}, \bar{X})$. Then there is a position $(\psi, \bar{x}, \bar{X}X, u)$, where $u \in \mathrm{children}_R(\mathrm{root}(R))$, such that the verifier has a winning strategy in $\mathcal{MC}(\mathrm{subtree}_R(u), \psi, \bar{x}, \bar{X}X)$ where $\mathrm{subtree}_R(u) = \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C}D)$ for some $D \subseteq A$. By the induction hypothesis, the verifier has a winning strategy in $\mathcal{MC}(F', \psi, \bar{x}, \bar{X}X)$, where $F' = \mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C}D) \in \mathrm{subtrees}(F)$. The verifier can therefore win $\mathcal{MC}(F, \varphi, \bar{x}, \bar{X})$ by choosing a position $(\psi, \bar{x}, \bar{X}X, \mathrm{root}(F'))$, which implies the claim.

If $\varphi = \forall x \psi(x)$, and the verifier has a winning strategy in one of the games, say in $\mathcal{MC}(R, \varphi, \bar{x}, \bar{X})$, consider a move of the falsifier to a position $(\psi, \bar{x}x, \bar{X}, u)$ in $\mathcal{MC}(F, \varphi, \bar{x}, \bar{X})$, where $u = \mathrm{root}(\mathrm{FC}_q(\mathscr{A}, \bar{c}d, \bar{C}))$ for some $d \in A$. Let $R' = \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C})$ be a subtree of the root of R. The verifier has a winning strategy in the game $\mathcal{MC}(R', \psi, \bar{x}x, \bar{X})$, and therefore, by the induction hypothesis, in

$\mathcal{MC}(\mathrm{FC}_q(\mathscr{A}, \bar{c}d, \bar{C}), \psi, \bar{x}x, \bar{X})$.

The remaining cases follow analogously.

□

From Lemmas 1, 2, and 3, we obtain the important fact that reduced characteristic trees are in fact equivalent to their full counterparts and characterize the equivalence relation $\equiv_q^{\mathrm{MSO}}$.

Corollary 1. *Let $\mathscr{A}$ and $\mathscr{B}$ be τ-structures and $q \in \mathbf{N}$. Then $\mathrm{RC}_q(\mathscr{A}) = \mathrm{RC}_q(\mathscr{B})$ iff $\mathscr{A} \equiv_q^{\mathrm{MSO}} \mathscr{B}$.*

The next lemma shows that reduced characteristic trees have small size. For $i \in \mathbf{N}$, we define $\exp^{(i)}(\cdot)$ as: $\exp^{(0)}(x) = x$, $\exp^{(1)}(x) = 2^x$ and $\exp^{(i)}(x) = 2^{2\exp^{(i-1)}(x)}$ for $i \geq 2$.

Lemma 4. *Let $\mathscr{A}$ be a τ-structure with universe A such that each relation symbol in τ has arity at most r, and $q \in \mathbf{N}$. Then the number of reduced characteristic trees $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ for all possible choices of $\bar{c}, \bar{C}$ is at most $\exp^{(q+1)}(|\tau| \cdot q^r + q\log q + q^2)$. The size of a reduced characteristic tree $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ is at most $(\exp^{(q)}(|\tau| \cdot q^r + q\log q + q^2))^4$.*

Proof. For integers m, p let $N(\mathscr{A}, m, p)$ be the number of trees $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$, where $\bar{c} = c_1, \ldots, c_m \in A^m$ and $\bar{C} = C_1, \ldots, C_p$ with $C_i \subseteq A$, $1 \leq i \leq p$. Define

$$S(\mathscr{A}, m, p) = \max_{\bar{c}, \bar{C}} |\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})|,$$

where the maximum is taken over all strings $\bar{c}$ and $\bar{C}$ such that $|\bar{c}| = m$ and $|\bar{C}| = p$. Also define $f(\tau, q) = |\tau| \cdot q^r + q\log q + q^2$.

If $m+p = q$ then $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ has one node for all $\bar{c}, \bar{C}$ and $S(\mathscr{A}, m, p) = 1$. The number of distinct trees $N(\mathscr{A}, m, p)$, however, depends on the number of structures on a universe of size at most $m \leq q$ over a vocabulary with $|\tau|$ relation symbols each of arity at most r. The number of such structures is at most $2^{|\tau| \cdot q^r}$, and since there are at most $q^q \cdot 2^{q^2}$ vectors $\bar{c}, \bar{C}$ over the $m + p \leq q$ elements, we have that $N(\mathscr{A}, m, p) \leq 2^{f(\tau,q)} \leq \exp^{(1)}(f(\tau, q))$. If $m + p < q$ then the root of $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ can have as children any of the $N(\mathscr{A}, m + 1, p)$

reduced characteristic trees corresponding to point moves and $N(\mathscr{A}, m, p+1)$ trees corresponding to set moves. Hence $N(\mathscr{A}, m, p) \leq 2^{N(m+1,p)+N(m,p+1)}$. By induction hypothesis, each of $N(\mathscr{A}, m+1, p)$ and $N(\mathscr{A}, m, p+1)$ is at most $\exp^{(q-(m+p))}(f(\tau, q))$ and hence

$$N(\mathscr{A}, m, p) \leq 2^{2 \cdot \exp^{(q-(m+p))}(f(\tau,q))} = \exp^{(q-(m+p)+1)}(f(\tau, q)).$$

Hence $N(\mathscr{A}, 0, 0) \leq \exp^{(q+1)}(f(\tau, q))$ as claimed.

The size of a reduced characteristic tree is one if $m + p = q$. Otherwise

$$S(\mathscr{A}, m, p) \leq 1 + S(\mathscr{A}, m+1, p)N(\mathscr{A}, m+1, p) + S(\mathscr{A}, m, p+1)N(\mathscr{A}, m, p+1),$$

since any such tree consists of a single root vertex and at most $N(\mathscr{A}, m+1, p)$ trees (corresponding to point moves) each of size $S(\mathscr{A}, m+1, p)$ and at most $N(\mathscr{A}, m, p+1)$ trees (corresponding to set moves) of size $N(\mathscr{A}, m, p+1)$. By induction hypothesis, each of the terms $S(\mathscr{A}, m+1, p)$ and $S(\mathscr{A}, m, p+1)$ is at most $(\exp^{(q-(m+p+1))}(f(\tau, q)))^4$ and hence

$$S(\mathscr{A}, m, p) \leq 1 + 2\exp^{(q-(m+p))}(f(\tau, q)) \cdot (\exp^{(q-(m+p+1))}(f(\tau, q)))^4.$$

One can show that the right hand side of the above inequality is at most $(\exp^{(q-(m+p))}(f(\tau, q)))^4$, thereby proving the claimed size bound.

□

5. Constructing Characteristic Trees

In this section, we show how to construct reduced characteristic trees of depth q for a graph G of rankwidth t when given a t-labeled parse tree decomposition of G. A t-labeled graph may be represented as τ-structure where $\tau = \{E, L_1, \ldots, L_t\}$. The symbol E is a binary relation symbol representing the edge relation and L_i for $1 \leq i \leq t$ is a unary relation symbol representing the set of vertices with label i. In what follows, whenever we talk about a τ-structure $\mathscr{A}$, we mean a graph viewed as a structure over the vocabulary $\{E, L_1, \ldots, L_t\}$.

Lemma 5. *Let* $\mathscr{A}$ *be a* τ*-structure with* $|A| = 1$*. Let* $q \geq 0$ *and* $\bar{c} \in A^m$ *and* $\bar{C} = C_1, \ldots, C_p$ *with* $C_i \subseteq A$*,* $1 \leq i \leq p$*. Then* $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ *can be constructed in constant time for each fixed* q*.*

Proof. Note that, in this case, $\mathrm{FC}_q(\mathscr{A}, \bar{c}, \bar{C})$ has size at most $O((2^1 + 1)^q) = O(3^q)$. Hence for each fixed q, $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ can be constructed in constant time.

□

In what follows, we let $\mathscr{A}_1, \mathscr{A}_2$ and $\mathscr{A} = \mathscr{A}_1 \otimes \mathscr{A}_2$ be τ-structures, where $\otimes = \otimes[g|f_1, f_2]$ for t-relabelings g, f_1, and f_2. Recall that if $\mathscr{A} = \mathscr{A}_1 \otimes \mathscr{A}_2$, then we assume that A_1 and A_2 (the universes of $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively) are disjoint. Furthermore for a fixed constant $q \geq 0$, let m and p be nonnegative integers such that $m + p \leq q$, $\bar{c} = c_1, \ldots, c_m \in (A_1 \cup A_2)^m$ and $\bar{C} = C_1, \ldots, C_p$, where $C_j \subseteq A_1 \cup A_2$, $1 \leq j \leq p$. For $i \in \{1, 2\}$, we let $\bar{c}_i = c_{i,1}, \ldots, c_{i,m_i} = \bar{c}[A_i]$.

In the remainder of this section, we show how to construct $\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ given $\mathrm{RC}_q(\mathscr{A}_1, \bar{c}_1, \bar{C} \cap \bar{c}_1)$ and $\mathrm{RC}_q(\mathscr{A}_2, \bar{c}_2, \bar{C} \cap \bar{c}_2)$. For the construction, as will be clear later on, we need to know the order in which the elements in $\bar{c}_1$ and $\bar{c}_2$ appear in $\bar{c}$. This motivates us to define the notion of an *indicator vector* $\mathrm{ind}(A_1, A_2, \bar{c})$.

Definition 8. The *indicator vector* of $\bar{c} = c_1, \ldots, c_m$, denoted $\mathrm{ind}(A_1, A_2, \bar{c})$, is the vector $\bar{d} = d_1, \ldots, d_m$, such that for $i \in \{1, 2\}$ and all $1 \leq j \leq m$ it holds that $d_j = (i, k)$ iff c_j is the kth element in the vector $\bar{c}_i = \bar{c}[A_i]$. If $\bar{d} = d_1, \ldots, d_m$ and $(i, k) \in \{1, 2\} \times [m + 1]$, then we use $\bar{d}(i, k) = \bar{d} \cdot (i, k)$ to denote the vector $d_1, \ldots, d_{m+1}$, where $d_{m+1} = (i, k)$.

Example 1. *Let* $A_1 = \{a_1, a_2\}$*,* $A_2 = \{b_1, b_2, b_3, b_4\}$ *and let* $\bar{c}$ *be the string* $a_1 b_1 b_2 a_2 b_3 b_4 a_2 b_3 a_1$*. Then we get:*

$\bar{c}$	$=$	a_1	b_1	b_2	a_2	b_3	b_4	a_2	b_3	a_1
$\bar{c}[A_1]$	$=$	a_1			a_2			a_2		a_1
$\bar{c}[A_2]$	$=$		b_1	b_2		b_3	b_4		b_3	
$\mathrm{ind}(A_1, A_2, \bar{c})$	$=$	$(1,1)$	$(2,1)$	$(2,2)$	$(1,2)$	$(2,3)$	$(2,4)$	$(1,3)$	$(2,5)$	$(1,4)$

Given $\bar{c}[A_1]$, $\bar{c}[A_2]$, and $\bar{d} = d_1, \ldots, d_m = \mathrm{ind}(A_1, A_2, \bar{c})$, one can now reconstruct $\bar{c}$. For example, $c_8 = b_3$, since $d_8 = (2, 5)$, which tells us that c_8 is the fifth element in $\bar{c}_2$.

Constructing $R = \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$ when given $R_1 = \mathrm{RC}_q(\mathscr{A}_1, \bar{c}_1, \bar{C} \cap \bar{c}_1)$, $R_2 = \mathrm{RC}_q(\mathscr{A}_2, \bar{c}_2, \bar{C} \cap \bar{c}_2)$, and $\bar{d} = \mathrm{ind}(A_1, A_2, \bar{c})$ consists of the following two steps:

1. construct the label for $\mathrm{root}(R) = \mathrm{Ord}(\mathscr{A}, \bar{c}, \bar{C})$, and then
2. recursively construct its subtrees.

Since $\mathrm{Ord}(\mathscr{A}, \bar{c}) \cong \mathscr{A}[\bar{c}]$ and $\mathscr{A}_i[\bar{c}_i] \cong \mathrm{Ord}(\mathscr{A}_i, \bar{c}_i)$, one easily sees that

$$\mathrm{Ord}(\mathscr{A}, \bar{c}) \cong \mathrm{Ord}(\mathscr{A}_1, \bar{c}_1) \otimes \mathrm{Ord}(\mathscr{A}_2, \bar{c}_2).$$

For the first step, we therefore just need to rename elements in $\mathrm{Ord}(\mathscr{A}_1, \bar{c}_1) \otimes \mathrm{Ord}(\mathscr{A}_2, \bar{c}_2)$ in an appropriate way. The information on how elements are to be renamed is stored in the indicator vector $\bar{d}$ of $\bar{c}$. See Figure 3 for an example. The formal definition of the renaming operator $\otimes_{\bar{d}}$ and Lemma 6 are technical and may be skipped if the reader believes that one can construct $\mathrm{Ord}(\mathscr{A}, \bar{c})$ from $\mathrm{Ord}(\mathscr{A}_1, \bar{c}_1)$ and $\mathrm{Ord}(\mathscr{A}_2, \bar{c}_2)$ using $\bar{d}$.

Definition 9. For $i \in \{1, 2\}$, let $\mathrm{Ord}(A_i, \bar{c}_i, \bar{C} \cap A_i) = (\mathscr{H}_i, \bar{c}'_i, \bar{C}'_i)$. Let $m := |\bar{c}_1| + |\bar{c}_2|$ and for $i \in \{1, 2\}$ let $l_i = |\bar{c}_i|$ and $H_i := [l_i]/{\equiv_{\bar{c}_i}}$. Define a map $f \colon [m] \to H_1 \uplus H_2$ as follows: for all $1 \leq j \leq m$, let $f(j) = [k]_{\bar{c}_i}$ iff $d_j = (i, k)$. Then we define $\mathrm{Ord}(\mathscr{A}_1, \bar{c}[A_1], \bar{C} \cap A_1) \otimes_{\bar{d}} \mathrm{Ord}(\mathscr{A}_2, \bar{c}[A_2], \bar{C} \cap A_2)$ as

$$\mathrm{Ord}(\mathscr{H}_1 \otimes \mathscr{H}_2, f(1) \ldots f(m), \bar{C}'_1 \cup \bar{C}'_2).$$

Lemma 6. *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be τ-structures and let $\otimes = \otimes[g|f_1, f_2]$ for some t-relabelings g, f_1, f_2. Let $\bar{c} = c_1, \ldots, c_m \in (A_1 \cup A_2)^m$ and $\bar{C} = C_1, \ldots, C_p$, where $C_j \subseteq A_1 \cup A_2$ for $1 \leq j \leq p$. Also let $\bar{d} = \mathrm{ind}(A_1, A_2, \bar{c})$. Then*

$$\mathrm{Ord}(\mathscr{A}_1 \otimes \mathscr{A}_2, \bar{c}, \bar{C}) = \mathrm{Ord}(\mathscr{A}_1, \bar{c}[A_1], \bar{C} \cap A_1) \otimes_{\bar{d}} \mathrm{Ord}(\mathscr{A}_2, \bar{c}[A_2], \bar{C} \cap A_2).$$

Proof. For $i \in \{1, 2\}$, it holds

$$\mathrm{Ord}(\mathscr{A}_i, \bar{c}_i, \bar{C}_i) = (\mathscr{H}_i, \bar{c}'_i, \bar{C}'_i) \cong (\mathscr{A}_i[\bar{c}[A_i]], \bar{c}[A_i], \bar{C} \cap A_i),$$

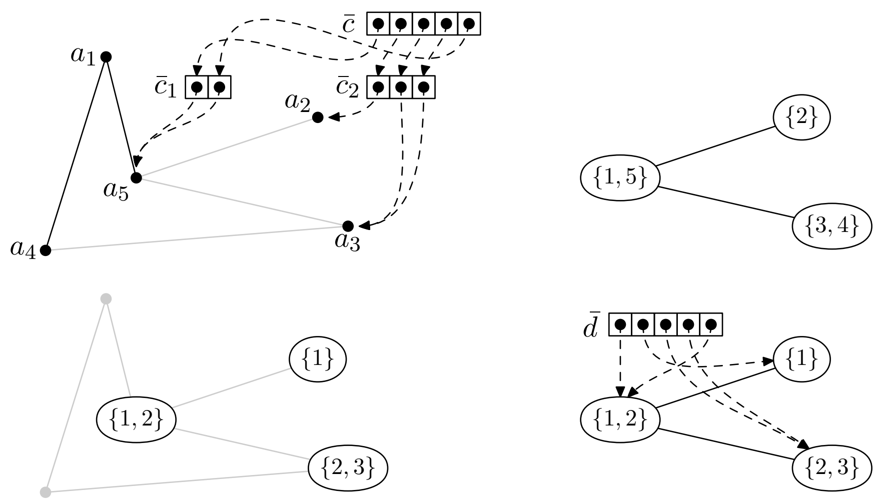

Figure 3: $\mathscr{G}_1$ and $\mathscr{G}_1$ depicted on the top left are graphs such that $\mathscr{G}_1 \oplus \mathscr{G}_2$ is the graph of Figure 1; the gray edges being those created by the t-labeled composition operator $\oplus$. For $\bar{c} = a_5 a_2 a_3 a_3 a_5$ and $\bar{c}_1 = \bar{c}[G_1]$, $\bar{c}_2 = \bar{c}[G_2]$ the ordered induced substructures $\mathscr{H}_1 = \mathrm{Ord}(\mathscr{G}_1, \bar{c}_1)$ and $\mathscr{H}_2 = \mathrm{Ord}(\mathscr{G}_1, \bar{c}_2)$ depicted in black on the bottom left. On these, we can take the t-labeled composition $\mathscr{H} = \mathscr{H}_1 \oplus \mathscr{H}_2$ and obtain the graph isomorphic to $\mathscr{G}_1[\bar{c}_1] \oplus \mathscr{G}_2[\bar{c}_2]$ on the bottom right. We can now use the vector $\bar{d} = (1,1)(2,1)(2,2)(2,3)(1,2)$ to rename vertices in $\mathscr{H}$ and obtain $\mathrm{Ord}(\mathscr{G}, \bar{c})$ depicted on the top right. Note that $\bar{c}$ and $\bar{d}$ essentially describe the same vertices.

where $h_i \colon c_{i,j} \mapsto [j]_{\bar{c}_i}$, $1 \leq j \leq m_i$ is the isomorphism of Definition 5 and $\bar{c}'_i = c'_{i,1}, \ldots, c'_{i,m_i} = h_i(1), \ldots, h_i(m_i) \in H_i^{m_i}$. Let $\mathscr{H} = \mathscr{H}_1 \otimes \mathscr{H}_2$ be the τ-structure with universe $H = H_1 \uplus H_2 = [m_1]/{\equiv_{\bar{c}_1}} \uplus [m_2]/{\equiv_{\bar{c}_2}}$, where we assume without loss of generality that H_1 and H_2 are disjoint (rename elements otherwise). We want to show that in the following diagram we have $\mathrm{Ord}(\mathscr{A}, c_1 \ldots c_m) = \mathrm{Ord}(\mathscr{H}, f(1) \ldots, f(m))$ (see also Figure 3 for a concrete example):

$$\begin{array}{ccccccc}
\mathscr{A}_1[\bar{c}_1] & \otimes & \mathscr{A}_2[\bar{c}_2] & = & \mathscr{A}[\bar{c}] & \cong & \mathrm{Ord}(\mathscr{A}, c_1 \ldots c_m) \\
\wr\!\| & & \wr\!\| & & & & \| \\
\mathscr{H}_1 & \otimes & \mathscr{H}_2 & = & \mathscr{H} & \cong & \mathrm{Ord}(\mathscr{H}, f(1) \ldots, f(m))
\end{array}$$

Informally, what the above diagram says is that if $\mathscr{A}_1[\bar{c}_1] \cong \mathscr{H}_1$ and $\mathscr{A}_2[\bar{c}_2] \cong \mathscr{H}_2$ then $\mathscr{A}_1[\bar{c}_1] \otimes \mathscr{A}_2[\bar{c}_2]$ and $\mathscr{H}_1 \otimes \mathscr{H}_2$ continue to be isomorphic. Therefore it does not matter whether we take the ordered induced structure of $\mathscr{A}[\bar{c}]$ or take the product of the ordered induced structures of $\mathscr{A}_1[\bar{c}_1]$ and $\mathscr{A}_2[\bar{c}_2]$. A formal proof of this follows.

For all $1 \leq j \leq m$, it holds

$$f(j) = \begin{cases} h_1(c_j) & \text{if } c_j \in A_1, \\ h_2(c_j) & \text{if } c_j \in A_2, \end{cases}$$

where $f\colon [m] \to H_1 \uplus H_2$ is the map from Definition 9. If $c_j \in A_i$, then $c_j = c_{i,k}$ for some $1 \leq k \leq m_i$ and therefore $d_j = (i,k)$. This implies $h_i(c_j) = [k]_{\bar{c}_i} = f(j)$ by Definition 5 and Definition 10. Therefore, $f(j_1) = f(j_2)$ iff $c_{j_1} = c_{j_2}$, which then implies lemma.

□

We now describe how to construct the subtrees of $R = \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})$. At this point, recall that each edge of R corresponds to either a point move or a set move in a model-checking game on $\mathscr{A}$ and that $|\bar{c}|$ $(|\bar{C}|)$ denotes the number of point (set) moves made thus far. Similarly the edges of $R_1 = \mathrm{RC}_q(\mathscr{A}_1, \bar{c}_1, \bar{C} \cap \bar{c}_1)$ and $R_2 = \mathrm{RC}_q(\mathscr{A}_2, \bar{c}_2, \bar{C} \cap \bar{c}_2)$ correspond to moves in the model-checking game on the substructures $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively. Recall also that $A = A_1 \uplus A_2$, where A, A_1, and A_2 are respectively the universes of $\mathscr{A}$, $\mathscr{A}_1$, and $\mathscr{A}_2$. If a player makes a point move in $\mathscr{A}$, then this corresponds to a point move in either $\mathscr{A}_1$ or in $\mathscr{A}_2$. Therefore in order to construct the subtrees of R corresponding to point moves, we take the cartesian product of the subtrees corresponding to point moves of R_1 ("choose an element in $\mathscr{A}_1$") with the tree R_2 ("no element in $\mathscr{A}_2$"), and vice versa. A set move in $\mathscr{A}$ may be thought of as the disjoint union of a set move in $\mathscr{A}_1$ and a set move in $\mathscr{A}_2$, since each $U \subseteq A$ may be written as $U_1 \uplus U_2$, where $U_1 \subseteq A_1$ and $U_2 \subseteq A_2$. Therefore in order to construct the subtrees of R corresponding to set moves, we take the cartesian product of the subtrees corresponding to set moves in R_1 with those in R_2.

We formalize the notion of the cartesian product of trees next.

Definition 10 (Tree Cross Product)**.** Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be τ-structures and let $\otimes = \otimes[g|f_1, f_2]$ for some t-relabelings g, f_1, f_2. For a fixed constant $q \geq 0$, let m and p be nonnegative integers such that $m+p \leq q$. Let $\bar{c} = c_1, \ldots, c_m \in (A_1 \cup A_2)^m$ and $\bar{C} = C_1, \ldots, C_p$, where $C_j \subseteq A_1 \cup A_2$, $1 \leq j \leq p$. For $i \in \{1,2\}$, let $\bar{c}_i = c_{i,1}, \ldots, c_{i,m_i} = \bar{c}[A_i]$, $q_i \geq q-m-p$, and $R_i = \mathrm{RC}_{q_i}(\mathscr{A}_i, \bar{c}_i, \bar{C} \cap A_i)$ with $\mathrm{root}(R_i) = (\mathscr{H}_i, \bar{c}_i', \bar{C}_i') = \mathrm{Ord}(A_i, \bar{c}_i, \bar{C} \cap A_i)$. We define the *tree cross product* of R_1 and R_2, $R = R_1 \times(q, \otimes, \bar{d})\ R_2$, to be a finite, rooted tree such that

- $\mathrm{root}(R) = \mathrm{root}(R_1) \otimes_{\bar{d}} \mathrm{root}(R_2)$, and
- if $m+p+1 \leq q$, then $\mathrm{subtrees}(R) = S_1 \cup S_2$, where

$$
\begin{aligned}
S_1 = \big\{ & \mathrm{subtree}_{R_1}(u_1) \times(q, \otimes, \bar{d} \cdot (1, m_1+1))\ R_2 \ \big| \\
& u_1 = (\mathscr{H}_1', \bar{c}_1' c, \bar{C}_1') \in \mathrm{children}_{R_1}(\mathrm{root}(R_1)) \big\} \cup \\
\big\{ & R_1 \times(q, \otimes, \bar{d} \cdot (2, m_2+1))\ \mathrm{subtree}_{R_2}(u_2) \ \big| \\
& u_2 = (\mathscr{H}_2', \bar{c}_2' c, \bar{C}_2') \in \mathrm{children}_{R_2}(\mathrm{root}(R_2)) \big\}
\end{aligned}
$$

  and

$$
\begin{aligned}
S_2 = \big\{ & \mathrm{subtree}_{R_1}(u_1) \times(q, \otimes, \bar{d})\ \mathrm{subtree}_{R_2}(u_2) \ \big| \\
& u_i = (\mathscr{H}_i', \bar{c}_i', \bar{C}_i' D_i) \in \mathrm{children}_{R_i}(\mathrm{root}(R_i)), 1 \leq i \leq 2 \big\}.
\end{aligned}
$$

We now show that $R = R_1 \times(q, \otimes, \bar{d})\ R_2$, where $R = \mathrm{RC}_q(\mathscr{A}_1 \otimes \mathscr{A}_2, \bar{c}, \bar{C})$ and $R_i = \mathrm{RC}_{q_i}(\mathscr{A}_i, \bar{c}_i, \bar{C} \cap A_i)$.

Lemma 7. *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be τ-structures and let $\otimes = \otimes[g|f_1, f_2]$ for some t-relabelings g, f_1, f_2. For nonnegative integers q, m, p with $m+p \leq q$, let $\bar{c} = c_1, \ldots, c_m \in (A_1 \cup A_2)^m$ and $\bar{C} = C_1, \ldots, C_p$, where $C_j \subseteq A_1 \cup A_2$ for $1 \leq j \leq p$. Also let $\bar{d} = \mathrm{ind}(A_1, A_2, \bar{c})$ and for $1 \leq i \leq 2$ let $q_i \geq q-m-p$. Then*

$$
\mathrm{RC}_q(\mathscr{A}_1 \otimes \mathscr{A}_2, \bar{c}, \bar{C}) = \mathrm{RC}_{q_1}(\mathscr{A}_1, \bar{c}_1, \bar{C} \cap A_1) \times(q, \otimes, \bar{d})\ \mathrm{RC}_{q_2}(\mathscr{A}_2, \bar{c}_2, \bar{C} \cap A_2).
$$

Proof. The proof is an induction over $q-m-p$. By Lemma 6,

$$
\mathrm{root}(\mathrm{RC}_q(\mathscr{A}_1 \otimes \mathscr{A}_2, \bar{c}, \bar{C})) = \mathrm{root}(R_1) \otimes_{\bar{d}} \mathrm{root}(R_2).
$$

If $q-m-p=0$, then $\mathrm{RC}_q(\mathscr{A}_1 \otimes \mathscr{A}_2, \bar{c}, \bar{C})$ consists of a single root node and the lemma holds. Otherwise, the set of subtrees is by definition

$$\begin{aligned}\text{subtrees}(\mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C})) = \big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \mid d \in A \,\big\} \cup \\ \big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C}D) \mid D \subseteq A \,\big\}.\end{aligned}$$

Here, by the induction hypothesis

$$\begin{aligned}&\big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \mid d \in A \,\big\} \\ &= \big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \mid d \in A_1 \,\big\} \cup \big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}d, \bar{C}) \mid d \in A_2 \,\big\} \\ &\overset{\text{i.h.}}{=} \big\{\, \mathrm{RC}_q(\mathscr{A}_1, \bar{c}[A_1]d, \bar{C} \cap A_1) \times(q, \otimes, \bar{d} \cdot (1, m_1+1))\ R_2 \mid d \in A_1 \,\big\} \cup \\ &\quad \big\{\, R_1 \times(q, \otimes, \bar{d} \cdot (2, m_2+1))\ \mathrm{RC}_q(\mathscr{A}_2, \bar{c}[A_2]d, \bar{C} \cap A_2) \mid d \in A_2 \,\big\} \\ &= S_1\end{aligned}$$

and, similarly,

$$\begin{aligned}&\big\{\, \mathrm{RC}_q(\mathscr{A}, \bar{c}, \bar{C}D) \mid D \subseteq A \,\big\} \\ &\overset{\text{i.h.}}{=} \big\{\, \mathrm{RC}_q(\mathscr{A}_1, \bar{c}[A_1], \bar{C}D \cap A_1) \times(q, \otimes, \bar{d})\ \mathrm{RC}_q(\mathscr{A}_2, \bar{c}[A_2], \bar{C}D \cap A_2) \;\big| \\ &\qquad D \in U \,\big\} \\ &= S_2.\end{aligned}$$

This concludes the proof.

□

Lemma 8. *Given R_1 and R_2, the tree cross product $R_1 \times(q, \otimes, \bar{d})\ R_2$ can be computed time poly$(|R_1|, |R_2|)$, where $|R_i|$ denotes the number of nodes in R_i.*

Proof. An algorithm computing $R_1 \times(q, \otimes, \bar{d})\ R_2$ may recursively traverse both trees top-down. For each pair of subtrees R_1' and R_2' of R_1 and R_2, the algorithm has to be called only once. The number of recursive calls is therefore bounded by $|R_1| \cdot |R_2|$ and each recursive call takes time dependent on q and the signature τ, and hence on the rankwidth t, only.

□

We now finally prove the Main Theorem.

The Main Theorem. [7, 12] *Let* φ *be an* MSO_1*-formula with* $\mathrm{qr}(\varphi) \leq q$. *There is an algorithm that takes as input a* t*-labeled parse tree decomposition* T *of a graph* G *and decides whether* $G \models \varphi$ *in time* $O(f(q,t) \cdot |T|)$, *where* f *is some computable function and* $|T|$ *is the number of nodes in* T.

Proof. It is no loss of generality to assume that G has at least one vertex. Otherwise deciding whether $G \models \varphi$ takes constant time. By Lemmas 1, 2 and 3, to prove that $G \models \varphi$ it is sufficient to show that the verifier has a winning strategy in the model checking game $\mathcal{MC}(\mathrm{RC}_q(G), \varphi, \epsilon, \epsilon)$. By Lemma 4, the size of the reduced characteristic tree $\mathrm{RC}_q(G)$ of a t-labeled graph is at most $f_1(q,t)$ for some computable function f_1 of q and t alone. By Lemma 8, the time taken to combine two reduced characteristic trees of size $f_1(q,t)$ is $f(q,t) = \mathrm{poly}(f_1(q,t))$.

We claim that the total time taken to construct $\mathrm{RC}_q(G)$ from its parse tree decomposition T is $O(f(q,t) \cdot |T|)$. The proof is by an induction on $|T|$. By Lemma 5, the claim holds when $|T| = 1$. Suppose that $\bar{G} = \bar{G}_1 \otimes [g|h_1,h_2]\, \bar{G}_2$, where g, h_1, h_2 are t-relabelings and let T_1 and T_2 be parse trees of $\bar{G}_1$ and $\bar{G}_2$, respectively. Then $|T| = |T_1| + |T_2| + 1$, where T is a parse tree of $\bar{G}$. By induction hypothesis, one can construct the reduced characteristic trees $\mathrm{RC}_q(G_1)$ and $\mathrm{RC}_q(G_2)$ in times $O(f(q,t) \cdot |T_1|)$ and $O(f(q,t) \cdot |T_2|)$, respectively. By Lemma 7, one can indeed construct $\mathrm{RC}_q(G)$ given $\mathrm{RC}_q(G_1)$, $\mathrm{RC}_q(G_2)$ and $\bar{d} = \varepsilon$. By using Lemma 8, the time taken to construct $\mathrm{RC}_q(G)$ is

$$O(f(q,t) + f(q,t) \cdot |T_1| + f(q,t) \cdot |T_2|) = O(f(q,t) \cdot |T|),$$

thereby proving the claim.

In order to check whether the verifier has a winning strategy in the model checking game $\mathcal{MC}(\mathrm{RC}_q(G), \varphi, \epsilon, \epsilon)$, one can use a very simple recursive algorithm (see also [14]). A position $p = (\psi, \bar{x}, \bar{X}, u)$ of the model checking game can be identified with a call of the algorithm with arguments p. If ψ is universal, then the algorithm recursively checks whether the verifier has a winning strategy from all positions u' that are reachable from u in the model checking game. If otherwise ψ is existential, then the algorithm checks whether there is one subsequent position in the game from which the verifier has a winning strategy. This algorithm visits each node of the reduced characteristic tree $\mathrm{RC}_q(G)$ at most once. Therefore the time taken to decide whether $G \models \varphi$ is $O(f_1(q,t) + f(q,t) \cdot |T|) = O(f(q,t) \cdot |T|)$, as claimed.

□

6. Discussion and Conclusion

The proof of the Main Theorem shows that deciding whether a graph models an MSO_1-sentence is linear-time doable if the rankwidth of the graph is bounded. The theorem by Courcelle et al. [7] says something stronger: one can compute the *optimal* solution to a linear optimization problem expressible in MSO_1 in linear time for graphs of bounded rankwidth. In its simplest form, a *linear optimization problem* in MSO_1 is a tuple

$$(\varphi(X_1, \ldots, X_l), a_1, \ldots, a_l, \mathrm{opt}),$$

where $\varphi(X_1, \ldots, X_l)$ is an MSO_1-formula with the free set variables $X_1, \ldots, X_l$, $\bar{a} = a_1, \ldots, a_l \in \mathbf{Z}^l$, and opt is either max or min. The objective is, given an input graph G, to find $(U_1, \ldots, U_l) \subseteq V(G)^l$ such that $G \models \varphi[X_1/U_1, \ldots, X_l/U_l]$ and $\sum_{i=1}^{l} a_i |U_i|$ is optimized (maximized or minimized).

One can use the techniques outlined in this paper to prove the stronger statement by first constructing reduced characteristic trees $\mathrm{RC}_q(G, \varepsilon, U_1, \ldots, U_l)$, of which there are only a function of q and l. All that remains to do is simulate the model checking game on each of the reduced characteristic trees and output the tuple $(U_1, \ldots, U_l)$ for which there is a winning strategy and $\sum_{i=1}^{l} a_i |U_i|$ is optimized.

An interesting question is whether the Main Theorem can be extended to MSO_2 formulas (with edge set quantifications) as can be done in Courcelle's Theorem on graphs of bounded treewidth [4]. In this context, recall that P_1 and NP_1 denote, respectively, the class of languages over a single letter (tally languages) that are in P and NP. Clearly P = NP implies $\mathrm{P}_1 = \mathrm{NP}_1$ but the other direction is not known. What is known is that $P_1 = NP_1$ if and only if EXPTIME = NEXPTIME [3, 17]. It was shown in [7] that if $P_1 \neq NP_1$ then there is an MSO_2-definable decision problem over the class of cliques that is not solvable in polynomial time. Since cliques have rankwidth one, this result illustrates the difficulty of extending the Main Theorem for MSO_2. Intuitively, the reason why our approach would fail for MSO_2 formulas is as follows: The operation $\bar{G}_1 \otimes \bar{G}_2$ in the parse tree "creates" an unbounded number m of edges between $\bar{G}_1$ and $\bar{G}_2$ for which there are 2^m edge-subsets to be considered. It does not seem possible to enhance the model-checking game with respect to these edge-subsets within polynomial time.

On the positive side, the results of this paper naturally extend to directed graphs and birankwidth. This allows us to conclude that any decision or optimization problem on directed graphs expressible in MSO_1 is linear-time solvable on graphs of bounded birankwidth [7, 20]. Finally, the game-theoretic approach has already been used to prove Courcelle's result for treewidth [4, 1, 9] with an emphasis on practical implementability [21].

References

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

[2] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays.* A.K. Peters, 1982.

[3] R.V. Book. Tally languages and complexity classes. *Information and Control*, 26:186–194, 1974.

[4] B. Courcelle. The monadic second order theory of Graphs I: Recognisable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[5] B. Courcelle. Monadic second-order definable graph transductions: A survey. *Theor. Comput. Sci.*, 126(1):53–75, 1994.

[6] B. Courcelle and M. M. Kante. Graph operations characterizing rank-width and balanced graph expressions. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, number 4769 in Lecture Notes in Computer Science, pages 66–75. Springer, 2007.

[7] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique Width. *Theory of Computing Systems*, 33:125–150, 2000.

[8] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[9] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109(1-2):49–82, 1993.

[10] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer, 1999.

[11] S. Feferman and R. Vaught. The first order properties of algebraic systems. *Fund. Math*, 47:57–103, 1959.

[12] R. Ganian and P. Hliněný. On parse trees and Myhill–Nerode–type tools for handling graphs of bounded rank-width. *Disc. App. Math.*, 158(7):851–867, 2010.

[13] R. Ganian, P. Hliněný, and J. Obdržálek. Unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. Submitted, 2009.

[14] E. Grädel. Finite model theory and descriptive complexity. In *Finite Model Theory and Its Applications*, pages 125–230. Springer, 2007.

[15] Y. Gurevich. Monadic second-order theories. In Solomon Feferman Jon Barwise, editor, *Model-Theoretic Logics*, pages 479–506. Springer-Verlag, 1985.

[16] Yuri Gurevich. Modest Theory of Short Chains. I. *J. Symb. Log.*, 44(4):481–490, 1979.

[17] J. Hartmanis. On sparse sets in NP−P. *Information Processing Letters*, 16:55–60, 1983.

[18] J. Hintikka. *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic.* Clarendon Press, 1973.

[19] P. Hliněný and S. Oum. Finding branch-decomposition and rank-decomposition. *SIAM Journal on Computing*, 38:1012–1032, 2008.

[20] M. M. Kante. The rankwidth of directed graphs. Preprint. Available at: `http://arxiv.org/abs/0709.1433`, 2007.

[21] J. Kneis, A. Langer, and P. Rossmanith. Courcelle's Theorem – a game-theoretic approach, 2011. Accepted to Discrete Optimization, doi `10.1016/j.disopt.2011.06.001`.

[22] A. Langer, P. Rossmanith, and S. Sikdar. Linear-time algorithms for graphs of bounded rankwidth: A fresh look using game theory - (extended abstract). In *TAMC'11*, volume 6648 of *LNCS*, pages 505–516. Springer, 2011.

[23] S. Oum. *Graphs of Bounded Rankwidth.* PhD thesis, Princeton University, 2005.

[24] S. Oum and P. D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory Series B*, 96(4):514–528, 2006.