THE LOGIC IN COMPUTER SCIENCE COLUMN

by

Yuri GUREVICH[1]

Tree-decompositions and the model-checking problem

Jörg Flum
Albert-Ludwigs-Universität Freiburg
79104 Freiburg, Germany
flum@ruf.uni-freiburg.de

1 Introduction

The notion of tree-decomposition (as the related one of tree-width) is of great importance in the theory of graphs. It has been introduced by Robertson and Seymour [27] (and, independently, by Halin [23]) who used it systematically in their analysis of graphs. Tree-decompositions have turned out to be a valuable tool in various areas of logic and computer science (in fact, Ajtai and Gurevich reinvented this notion in [1]). In logic, the concept of tree-decomposition has been used most systematically for the model-checking problem. This problem is one of the main issues of descriptive complexity; it analyzes the complexity of problems of the form:

> Decide if a given structure is a model of a given sentence (of a fixed logic).

The model-checking problem arises very naturally in various areas of computer science. Let us see two examples.

In the *computer-aided verification* area the basic problem is the model-checking problem: does a given program (the "structure") satisfy a desired property (the "sentence") expressed in some specification language (the "logic"). And in *database theory*, (relational) databases are viewed as structures and (Boolean) queries to the databases as sentences; thus, the query evaluation problem is the model-checking problem for the corresponding query language (the "logic"). We mostly deal with logics relevant for this second area and do not consider logics like modal and temporal logics relevant for computer-aided verification.

[1] http://research.microsoft.com/ gurevich

Already in 1973, Stockmeyer [29] showed that the model-checking problem for first-order logic is PSPACE-complete. Since then, various results handling this high worst-case complexity have been obtained. Let us briefly recall two of the pioneering results due to Yannakakis and Courcelle, respectively (later we give precise statements and explain the notions):

- The model-checking problem for acyclic conjunctive sentences can be solved in polynomial time.
- For every sentence of monadic second-order logic, the model-checking problem on a class of structures of bounded tree-width can be solved in linear time.

The concept of acyclic conjunctive sentences appearing in Yannakakis' theorem is defined via the notion of tree-decomposition; Courcelle restricts the admitted sructures to a class of structures of bounded tree-width. Thus, in both cases, the concept of tree-decomposition yields the frame that allows one to formulate and prove the corresponding result. Of course, it is conceivable that there is a more far-reaching concept than that of tree-decomposition and that the theorems above can be extended to this broader framework. However, results obtained quite recently show that sometimes the concept of tree-decomposition allows the separation of "manageable" instances of the model-checking problem from "unmanageable" ones. Therefore, these recent results suggest that the concept of tree-decomposition really is an essential concept for descriptive complexity, and thus for logic in computer science.

The purpose of this article is to survey some of the results on model-checking based on the notion of tree-decomposition. Various papers surveying the graph-theoretic, algorithmic, or logical relevance of this concept already exist. We just mention two, which appeared in this Bulletin: [3, 8].[2]

2 Preliminaries

A *vocabulary* τ is a finite set of relation symbols. A τ*-structure* $\mathcal{A}$ consists of a non-empty set A, called the *universe* of $\mathcal{A}$, and a relation $R^{\mathcal{A}} \subseteq A^r$ for each r-ary relation symbol $R \in \tau$. We only consider finite structures. If $\mathcal{A}$ is a structure and $B \subseteq A$ non-empty, then $\langle B \rangle^{\mathcal{A}}$ denotes the substructure induced by $\mathcal{A}$ on B. We denote by STR the class of all structures.

We consider *graphs* as $\{E\}$-structures $\mathcal{G} = (G, E^{\mathcal{G}})$, where $E^{\mathcal{G}}$ is an irreflexive and symmetric binary relation. A *directed graph* is an $\{E\}$-structure $\mathcal{G} = (G, E^{\mathcal{G}})$ where $E^{\mathcal{G}} \subseteq G \times G$ is arbitrary. Then, its *underlying graph* $\mathcal{G}^u$ is given by $\mathcal{G}^u := (G, \{(a,b) \mid a \neq b, (E^{\mathcal{G}}ab \text{ or } E^{\mathcal{G}}ba)\}$.

[2] I thank Joe Flenner for some helpful comments.

We assume that the reader is familiar with first-order logic and monadic second-order logic. FO and MSO denote the class of first-order formulas and the class of monadic second-order formulas, respectively.

We use lowercase letters $x, y, \dots$ to denote first-order variables and uppercase letters $X, Y, \dots$ to denote monadic second-order variables. The set of free variables of a formula φ is denoted by free(φ). We write $\varphi(X_1, \dots, X_l, x_1, \dots, x_m)$ to indicate that $\text{free}(\varphi) = \{X_1, \dots, X_l, x_1, \dots, x_m\}$. However, if we just write φ this does not mean that φ is a sentence, i.e., that $\text{free}(\varphi) = \emptyset$.

We often denote tuples $(a_1, \dots, a_k)$ of elements of a set A by $\bar{a}$, and we write $\bar{a} \in A$ instead of $\bar{a} \in A^k$. Similarly, we denote tuples $(B_1, \dots, B_k)$ of subsets of A by $\bar{B}$ and write $\bar{B} \subseteq A$ instead of $\bar{B} \in (\text{Pow}(A))^k$. We denote tuples of variables by $\bar{x}$ or $\bar{X}$.

For a structure $\mathcal{A}$ and a formula $\varphi(X_1, \dots, X_l, x_1, \dots, x_m)$ we let

$$\varphi(\mathcal{A}) := \left\{ (B_1, \dots, B_l, a_1, \dots, a_m) \mid \bar{B} \subseteq A,\ \bar{a} \in A, \text{ and } \mathcal{A} \models \varphi(\bar{B}, \bar{a}) \right\}$$

be the set of assignments satisfying φ in $\mathcal{A}$.

The *size* $\|\mathcal{A}\|$ of a structure $\mathcal{A}$ and the size $\|\varphi\|$ of a formula φ are the lengths of a reasonable encoding of $\mathcal{A}$ and of φ, respectively.

Let C be a class of structures and L a logic. By MC(C, L) we denote the *model-checking problem for structures in* C *and sentences in* L:

Input:	Structure $\mathcal{A} \in$ C, sentence φ in L.
Problem:	Is $\mathcal{A}$ a model of φ?

MC(C, L) $\in$ COMPL, where COMPL is a complexity class, means that there is an algorithm in COMPL solving MC(C, L) (its complexity being measured as a function of both the size of the input structure and the size of the input sentence (*combined complexity*)).

D-MC(C, L) $\in$ COMPL means that the *database complexity* of the model-checking problem for C and L is in COMPL, that is, that

> for all sentences $\varphi \in$ L there is an algorithm in COMPL solving the problem
>
> | | |
> |---|---|
> | *Input:* | Struture $\mathcal{A} \in$ C. |
> | *Problem:* | Is $\mathcal{A}$ a model of φ? |

(i.e., for every sentence in L there is a corresponding algorithm whose complexity is measured as a function of the size of the input structure).

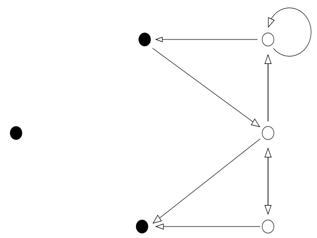

Figure 1

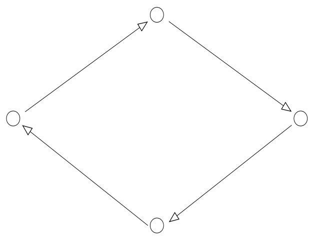

Figure 2

3 Trees

A *tree* $\mathcal{T}$ is a connected acyclic graph. Often we tacitly fix an (arbitrary) *root* $r^{\mathcal{T}} \in T$ in a tree $\mathcal{T}$. Then we have a natural partial order $\leq^{\mathcal{T}}$ on T, which is defined by

$$t \leq^{\mathcal{T}} u \iff t \text{ appears on the (unique) path from } r^{\mathcal{T}} \text{ to } u.$$

We say that u is a *child* of t or t is the *parent* of u if $(t, u) \in E^{\mathcal{T}}$ and $t \leq^{\mathcal{T}} u$. For every $t \in T$ we let $\mathcal{T}_t := \langle\{u \mid t \leq^{\mathcal{T}} u\}\rangle^{\mathcal{T}}$ be the subtree rooted at t. A tree $\mathcal{T}$ is *binary* if every vertex has either 0 or 2 children. A *directed tree* is a directed graph whose underlying graph is a tree.

Our starting point is the observation that some hard problems for graphs are easy for trees. For example, Hamiltonicity is an NP-complete problem for graphs, but a tree has no Hamiltonian cycle, since it has no cycle at all. Similarly, 3-colorability is an NP-complete for graphs, but every tree is 3-colorable (even 2-colorable, just alternate colors from level to level).

We present a further example: A set K of vertices in a directed graph $\mathcal{G}$ is a *kernel* if

$$\forall x, y \in K: \text{ not } E^G xy \quad \text{and} \quad \forall x \in (G \setminus K)\, \exists y \in K: \ E^G xy.$$

The directed graph in Figure 1 has a unique kernel consisting of the marked points. One easily verifies that the directed graph in Figure 2 has two kernels. The problem to decide whether a given directed graph has a kernel is NP-complete (cf. [6]). The situation is different for directed trees: we present an algorithm that works in linear time. Its underlying idea will then be generalized in order to prove some further theorems.

Let $\mathcal{T} = (T, E^{\mathcal{T}})$ be a directed tree and $\mathcal{T}^u$ its underlying tree. For $a \in T$ let $\mathcal{T}_a = (T_a, E_a)$ be the directed tree induced by $\mathcal{T}$ on $T_a := \{b \mid a \leq^{\mathcal{T}^u} b\}$ (i.e. induced by $\mathcal{T}$ on the universe of the subtree of $\mathcal{T}^u$ rooted at a). If a is distinct from the root, let parent(a) denote the parent of a in $\mathcal{T}^u$.

Essentially, the algorithm proceeds as follows: From the leaves to the root, it checks, for every node a, whether there is a kernel of $\mathcal{T}_a$ containing a and whether there is a kernel of $\mathcal{T}_a$ not containing a.

More precisely, we associate with every vertex $a \in T$ a pair $(k_a, n_a) \in \{0,1\} \times \{0,1,2\}$ where

$$k_a := \begin{cases} 1 & \text{there is a kernel of } \mathcal{T}_a \text{ containing } a; \\ 0 & \text{otherwise;} \end{cases}$$

and

$$n_a := \begin{cases} 1 & \text{there is a kernel of } \mathcal{T}_a \text{ not containing } a; \\ 2 & \text{there is no kernel of } \mathcal{T}_a \text{ not containing } a, \\ & \text{but } E^{\mathcal{T}} a \operatorname{parent}(a) \text{ and there is a kernel of} \\ & (T_a \cup \{\operatorname{parent}(a)\}, E_a \cup \{(a, \operatorname{parent}(a))\}) \text{ not containing } a; \\ 0 & \text{otherwise.} \end{cases}$$

These pairs of numbers can easily be computed from the leaves to the root: If a is a leaf then

$$k_a = \begin{cases} 1 & \text{if not } E^{\mathcal{T}} aa \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad n_a = \begin{cases} 2 & \text{if } E^{\mathcal{T}} a \operatorname{parent}(a) \\ 0 & \text{otherwise.} \end{cases}$$

If a is not a leaf then

$$\begin{array}{lll} k_a = 1 & \text{iff} & \text{not } E^{\mathcal{T}} aa \text{ and for all children } b \text{ of } a \text{ (in } \mathcal{T}^u\text{): } n_b \neq 0; \\ n_a = 1 & \text{iff} & \text{for all children } b \text{ of } a \ (k_b + n_b > 0 \text{ and } (n_b = 2 \text{ implies } k_b = 1)) \\ & & \text{and there is a child } b \text{ of } a \text{ s.t. } (E^{\mathcal{T}} ab \text{ and } k_b = 1); \\ n_a = 2 & \text{iff} & \text{for all children } b \text{ of } a \ (k_b + n_b > 0 \text{ and } (n_b = 2 \text{ implies } k_b = 1)), \\ & & \text{there is no child } b \text{ of } a \text{ s.t. } (E^{\mathcal{T}} ab, \text{ and } k_b = 1), \text{ and } E^{\mathcal{T}} a \operatorname{parent}(a); \end{array}$$

(cf. Figure 3 and Figure 4). Now, the directed tree $\mathcal{T}$ has a kernel just in case $n_r + k_r > 0$ holds for the root r. In the positive case one can construct a kernel by traversing the tree again, now from the root to the leaves; altogether, the algorithm needs a time linear in the size of $\mathcal{T}$ to decide if $\mathcal{T}$ has a kernel and if so, to construct one.

Not every NP-complete problem for graphs has a feasible algorithm on trees, one of the best-known examples being the band-width optimization problem, which remains NP-complete for the class of trees [25]. It asks for the least k such that there is an ordering of the vertices such that in this ordering at most k elements are between the ends of any edge.

While the formulation of the band-width problem involves the existence of a binary relation (the ordering), the existence of a 3-coloring and the existence of a kernel can be expressed in monadic second-order logic MSO – the latter, for instance, by

$$\exists X (\forall x \forall y ((Xx \wedge Xy) \rightarrow \neg Exy) \wedge \forall x (\neg Xx \rightarrow \exists y (Xy \wedge Exy))).$$

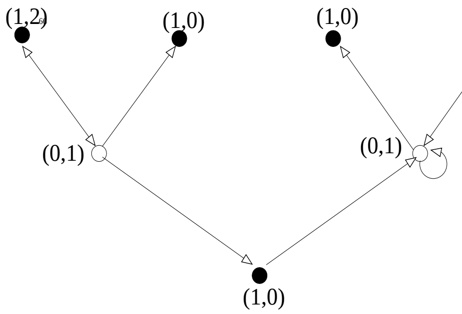

Figure 3

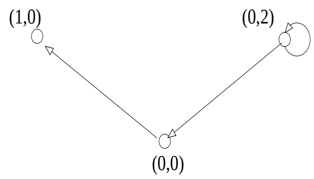

Figure 4

We denote by TREE the class of all trees. The positive results for trees mentioned above are special instances of the following theorem:

Theorem 3.1 ([30]) D-MC(TREE, MSO) $\in$ LINTIME, *that is, for every sentence* φ *of* MSO *there is an algorithm that given a tree* $\mathcal{T}$, *checks if* $\mathcal{T} \models \varphi$ *in time* $O(\|\mathcal{T}\|)$.[3]

Theorem 3.1 can be generalized from trees to tree-like structures. In section 5 we will sketch a proof of this generalization. In the main step of the original proof of 3.1, it was shown that the classes of trees axiomatizable in MSO are just those acceptable by a (deterministic) tree-automaton.

Tree-decompositions yield the right concept of tree-like structures. We recall the definition in the next section.

4 Tree-decompositions

Let $\mathcal{A}$ be a structure of vocabulary τ. $(\mathcal{T}, (X_t)_{t \in T})$ is a *tree-decomposition* of $\mathcal{A}$ (or, $(\mathcal{T}, (X_t)_{t \in T})$ is a decomposition of $\mathcal{A}$ along the tree $\mathcal{T}$), if

- $\mathcal{T}$ is a tree;
- For $a \in A$ the set $\{t \mid a \in X_t\}$ is non-empty and connected (in $\mathcal{T}$);
- For every $R \in \tau$ and every $\bar{a}$ with $R^{\mathcal{A}}\bar{a}$ there is $t \in T$ such that $\bar{a} \in X_t$.

The set X_t is called the *block* at t. The *width* of $(\mathcal{T}, (X_t)_{t \in T})$ is $\max\{|X_t| \mid t \in T\} - 1$.

The *tree-width* $\mathrm{tw}(\mathcal{A})$ of $\mathcal{A}$ is the minimal width over all possible tree-decompositions of $\mathcal{A}$. If C is a class of structures, then $\mathrm{tw}(\mathrm{C}) := \max\{\mathrm{tw}(\mathcal{A}) \mid \mathcal{A} \in \mathrm{C}\}$. And C is of *bounded tree-width*, if $\mathrm{tw}(\mathrm{C}) < \infty$. For $w \geq 1$, let STR_w denote the class of structures of tree-width $\leq w$.

[3] The result remains true for the class of labelled trees which "contains" the directed trees.

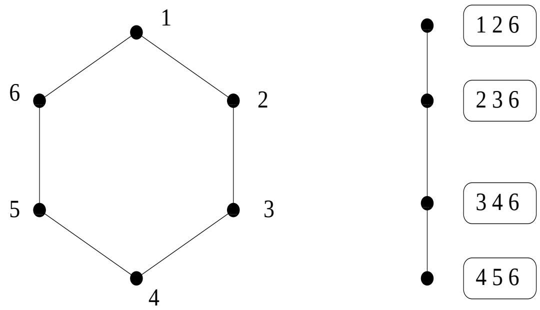

Figure 5

The tree-width of a graph $\mathcal{G}$ is a parameter that measures how close to a tree $\mathcal{G}$ is. Trees and forests have tree-width ≤ 1, cycles tree-width 2 (cf. Figure 5) and a clique of k vertices has tree-width $k-1$; thus, a clique has maximal tree-width, since $\mathrm{tw}(\mathcal{A}) \leq |A| - 1$ for every structure $\mathcal{A}$.

The *graph* $\mathcal{G}(\mathcal{A})$ of a structure $\mathcal{A}$ of vocabulary τ is the graph with universe A in which two elements $a \neq b$ are adjacent if there is an $R \in \tau$ and a tuple $\bar{a} \in R^{\mathcal{A}}$ such that both a and b occur in the tuple $\bar{a}$. One can verify that the tree-decompositions of $\mathcal{G}(\mathcal{A})$ are precisely the tree-decompositions of $\mathcal{A}$ and thus, $\mathrm{tw}(\mathcal{G}(\mathcal{A})) = \mathrm{tw}(\mathcal{A})$. So, in many contexts, it would be possible to restrict our attention to graphs, but we do this only if the formulations for graphs are more natural.

We mentioned in the introduction that the notion of tree-decomposition had been reinvented in logic and called *nostrum* by Ajtai and Gurevich (cf. [1], 8.1 and 8.2). Before turning to the model-checking problem, we just want to state the main property of tree-decompositions proved and applied in [1].

Let $(\mathcal{T}, (X_t)_{t\in T})$ be a tree-decomposition of $\mathcal{A}$. For $t \in T$ and t_1, t_2 in distinct connected components of $\mathcal{T} \setminus t$, we have:

$$(*) \qquad \forall a \in (X_{t_1} \setminus X_t)\, \forall b \in (X_{t_2} \setminus X_t)\, \forall R \in \tau: \ \text{not } R^{\mathcal{A}} \dots a \dots b \dots$$

i.e., the distance of $a \in (X_{t_1} \setminus X_t)$ and $b \in (X_{t_2} \setminus X_t)$ in the graph $\mathcal{G}(\mathcal{A})$ is at least two. Recall that a subset X of $\mathcal{A}$ is *d-scattered*, if the distance in $\mathcal{G}(\mathcal{A})$ between any two elements of X exceeds d. Thus, $\{a, b\}$ above is 1-scattered. The following lemma essentially says that sufficiently large structures of tree-width $< w$ have reasonably big d-scattered subsets, if you remove up to w appropriate elements:

Lemma 4.1 ([1]) *There is a function f such that for all $m, d, w \geq 1$ and all structures $\mathcal{A}$ of tree-width $< w$ with $|A| \geq f(m, d, w)$ there exists $S \subseteq A$, $|S| \leq w$, such that in $\langle A \setminus S\rangle^{\mathcal{A}}$ there is a d-scattered subset of cardinality m.*

Ajtai and Gurevich used this property of tree-decompositions to prove the main result of their paper, namely:

> Every DATALOG query expressible in first-order logic is bounded.

5 The model-checking problem for tree-like structures

In the Introduction we already mentioned our belief that the model-checking problem is the area of logic where the notion of tree-decomposition has been used most systematically. In this section and in section 6 we will present and discuss some of the results obtained.

5.1 Monadic second-order logic

By the following theorem, 3-colorability and the existence of a kernel can be checked quickly for classes of graphs of bounded tree-width.

Theorem 5.1 ([9]) *Let* C *be a class of structures of bounded tree-width. Then,* D-MC(C, MSO) $\in$ LINTIME.

We sketch a proof (cf. [21]). It builds on an idea already present in our treatment of the kernel problem for directed trees. This idea will be further generalized later. Moreover, it uses the following result showing that tree-decompositions can be computed quickly:

Theorem 5.2 ([4]) *There is an algorithm that, given a structure $\mathcal{A}$ and $w \geq 1$, computes a tree-decomposition of $\mathcal{A}$ of width at most w if* $\mathrm{tw}(\mathcal{A}) \leq w$*, or rejects if* $\mathrm{tw}(\mathcal{A}) > w$*. The algorithm needs time $O(2^{c \cdot w^3} \cdot \|\mathcal{A}\|)$.*

We come to the proof of Theorem 5.1: Fix an MSO-sentence φ, say of quantifier rank q. Set $w := \mathrm{tw}(\mathrm{C})$. Let $\mathcal{A}$ be a structure in C. First, (using the previous result) the desired algorithm computes a tree-decomposition $(\mathcal{T}, (X_t)_{t \in T})$ of tree-width $\leq w$. W.l.o.g. we assume that $\mathcal{T}$ is a binary tree.

For $t \in T$ let $Y_t := \bigcup_{s \geq t} X_s$. Define the sequence $\bar{a}^t := a_1 \ldots a_{w+1}$ such that $X_t = \{a_1, \ldots, a_{w+1}\}$. The *q-type* of $\bar{a}^t$ is the set of formulas of MSO of quantifier rank $\leq q$ satisfied by $\bar{a}^t$ in $\langle Y_t \rangle^{\mathcal{A}}$. (As already mentioned, the algorithm proceeds as the corresponding one for the kernel problem for directed trees: for $t \in T$, the role of the tree rooted at t is now taken over by Y_t, the role of t by $\bar{a}^t$ and the role of (k_t, n_t) by the q-type of $\bar{a}^t$.)

Using a tool of model theory, the Ehrenfeucht-Fraïssé games for MSO (e.g., cf. [12]), one can show that, for $t \in T$ with children u, v, the q-type of $\bar{a}^t$ only depends on the following data:

- the isomorphism type of the substructure induced by $\bar{a}^t$;
- the q-types of $\bar{a}^u$ and $\bar{a}^v$;
- the "intersection" of $\bar{a}^t$ and $\bar{a}^u$, and of $\bar{a}^t$ and $\bar{a}^v$.

The function giving these dependencies (for the fixed q) can be computed in advance. From the leaves to the root the algorithm we aim at calculates the q-type of any t.[4] From the q-type of the root it gets the information whether $\mathcal{A} \models \varphi$. Thus, the algorithm passes the tree once and performs at any node a bounded number of steps; altogether, the number of steps is $O(\|\mathcal{A}\|)$. □

For graphs of bounded tree-width one can also exhibit a concrete kernel (if there is at least one) in linear time:

Theorem 5.3 ([13]) *Let* C *be a class of bounded tree-width and* $\varphi(\bar{Y}, \bar{x})$ *an* MSO-*formula. There is an algorithm that, given* $\mathcal{A} \in$ C, *decides whether* $\varphi(\mathcal{A}) \neq \emptyset$ *and, in the positive case, computes* $B_1, \ldots, B_l \subseteq A$ *and* $a_1, \ldots, a_m \in A$ *such that*

$$\mathcal{A} \models \varphi(B_1, \ldots, B_l, a_1, \ldots, a_m)$$

in time $O(\|\mathcal{A}\|)$.

This result has recently been applied (cf. [20]) to show that, for every $k \geq 0$, there is a quadratic time algorithm deciding whether a given graph has crossing number at most k and, in the positive case, to compute a drawing of the graph in the plane with at most k crossings.

We already mentioned that the class of trees axiomatizable in MSO are just those accepted by so-called deterministic tree-automata. Using these deterministic automata one can extend the previous results and get fast algorithms computing, on classes of bounded tree-width, all satisfying assignments (i.e., solving the so-called *evaluation problem*):

Theorem 5.4 ([13]) *Let* C *be a class of structures of bounded tree-width and* $\varphi(\bar{Y}, \bar{x})$ *an* MSO-*formula. There is an algorithm solving*

Input:	*Structure* $\mathcal{A} \in$ C.
Problem:	*Compute* $\varphi(\mathcal{A})$.

in time $O(\|\mathcal{A}\| + \|\varphi(\mathcal{A})\|)$.

[4] Simple techniques from logic show that there is a finite set $\Phi(q, w)$ of MSO-formulas $\varphi(x_1, \ldots, x_{w+1})$ of quantifier rank $\leq q$ such that every MSO-formula $\psi(x_1, \ldots, x_{w+1})$ of quantifier rank $\leq q$ is equivalent to one in $\Phi(q, w)$; thus, we can assume that the q-types are finite objects.

Example 5.5 *Let $w \geq 1$. There is an algorithm that, given a graph $\mathcal{G}$ with* $\mathrm{tw}(\mathcal{G}) \leq w$*, lists all kernels of $\mathcal{G}$ in time $O(|\mathcal{G}| + \sum_{K \text{ kernel in } \mathcal{G}} |K|)$.*

We fix $w \geq 1$ and consider $\mathrm{MC}(\mathrm{STR}_w, \mathrm{MSO})$ (recall that STR_w denotes the class of structures of tree-width $\leq w$) . This problem – as most combined complexity model-checking problems – has a high complexity. In fact, an analysis of the proof of Courcelles's Theorem 5.1 given above reveals that $\mathcal{A} \models \varphi$ can be decided in time $\leq f(\|\varphi\|) \cdot \|\mathcal{A}\|$, where f is some (fast-growing) function that bounds the number of steps needed to calculate the dependencies of the q-types, where q is the quantifier rank of φ; note that φ now is part of the input and thus, these dependencies cannot be computed in advance. Therefore, the main factor contributing to the high complexity of $\mathrm{MC}(\mathrm{STR}_w, \mathrm{MSO})$ is the sentence φ. In database theory, (relational) databases are viewed as structures and queries to the databases as formulas. Queries are often short, mostly much shorter than the size of the databases. The so-called *parameterized complexity* tries to take care of this difference (cf. [11]). We denote by $\text{P-MC}(\mathrm{C}, \mathrm{L})$ the parameterized complexity of the model-checking problem for C and L, and define the parameterized complexity classes FPT and FPLIN just for the model-checking problem:

$\text{P-MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{FPT}$ – read: the model-checking for C and L is *fixed-parameter tractable* – if there is a computable function $f : \mathrm{L} \to \mathbb{N}$, a constant c, and an algorithm solving

Input:	Structure $\mathcal{A} \in \mathrm{C}$, sentence φ in L.
Problem:	$\mathcal{A} \models \varphi$?

in time $\leq f(\varphi) \cdot ||\mathcal{A}||^c$ (note that the constant c does not depend on φ).

We write $\text{P-MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{FPLIN}$, if c can be chosen to be 1.

Clearly,

$$\begin{array}{rcl} \mathrm{MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{PTIME} & \text{implies} & \text{P-MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{FPT}; \\ \text{P-MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{FPT} & \text{implies} & \text{D-MC}(\mathrm{C}, \mathrm{L}) \in \mathrm{PTIME}; \end{array}$$

thus, the fixed-parameter tractability of the model-checking problem is a notion lying between the feasibility of the "database model-checking problem" and the feasibility of the "combined model-checking problem".

As remarked above:

Corollary 5.6 *Let* C *be a class of bounded tree-width. Then,* $\text{P-MC}(\mathrm{C}, \mathrm{MSO}) \in \mathrm{FPLIN}$.

For classes of graphs closed under minors[5] the corollary is optimal, as shown by:

[5]A graph $\mathcal{H}$ is a *minor* of a graph $\mathcal{G}$, if $\mathcal{H}$ can be obtained from a subgraph of $\mathcal{G}$ by contracting edges.

Theorem 5.7 ([21]) *Assume* P $\neq$ NP. *Let* C *be a class of graphs closed under minors. Then, the following are equivalent:*

(i) P-MC(C, MSO) $\in$ FPLIN.

(ii) P-MC(C, MSO) $\in$ FPT.

(iii) tw(C) $< \infty$.

In view of Corollary 5.6 it suffices to show the implication (ii) $\Rightarrow$ (iii). Hence, let C be a class of graphs closed under minors and assume, by contradiction, that tw(C) $= \infty$. Then, by the Excluded Grid Theorem ([28]), C contains all planar graphs. But 3-colorability is NPTIME-complete on the class of planar graphs, contradicting (ii). $\square$

Thus, for monadic-second order logic, the notion of tree-width allows to characterize the fixed-parameter tractable instances of the model-checking problem.

5.2 First-order logic

For first-order logic we obtain a generalization of Courcelle's Theorem by applying Gaifman's Theorem [16]. Essentially, Gaifman's Theorem states that every first-order sentence only claims the existence of a certain number of pairwise disjoint balls whose centers satisfy some first-order property in (the substructure generated by) the balls. Therefore, for first-order sentences, we are not forced to require that the whole structure has bounded tree-width, but that the balls do. More precisely:

The *local tree-width* of $\mathcal{A}$ is the function $\mathrm{ltw}(\mathcal{A}) : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathrm{ltw}^{\mathcal{A}}(r) \quad := \quad \max\{\mathrm{tw}(\langle B_r(a)\rangle^{\mathcal{A}}) \mid a \in A\},$$

where $B_r(a)$ denotes the ball of radius r around a in the graph $\mathcal{G}(\mathcal{A})$ of $\mathcal{A}$. A class C of structures has *bounded local tree-width*, if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that

$$\text{for all } \mathcal{A} \in \mathrm{C} \text{ and all } r \geq 1 : \mathrm{ltw}^{\mathcal{A}}(r) \leq f(r).$$

As already indicated one gets:

Theorem 5.8 ([15]) *Let* C *be a class of structures of bounded local tree-width. Then,* P-MC(C, FO) $\in$ FPT.

Of course, every class of structures of bounded tree-width is of bounded local tree-width. Further examples of classes of bounded local tree-width are the class of graphs of bounded valence or the class of planar graphs.

We sketch the idea leading to a further generalization for first-order logic. For this purpose, we again look at the proof of Courcelle's Theorem: there, we essentially need that the first-order q-types of the blocks are computable quickly; so, at this stage, blocks of bounded local tree-width would suffice. Moreover, we do not need that the the sizes of the blocks X_t are bounded, but that the sizes of the intersections $X_t \cap X_s$ with $t, s \in T$, $t \neq s$, are bounded. A decomposition theorem of Grohe [19] based on deep graph theoretic results due to Robertson and Seymour says: If C is a class of graphs such that there is a graph that is not a minor of any graph in C, then we get such tree-decompositions for the members in C (compare [19] for the precise statement).

Theorem 5.9 ([14]) *Let* C *be a class of graphs such there exists a graph that is not a minor of a graph in* C. *Then* $\text{P-MC}(\text{C}, \text{FO}) \in \text{FPT}$.

In parameterized complexity theory the class FPT plays the role that PTIME does in classical complexity theory, and the role of NPTIME is taken over by a class denoted by $W[1]$. As with PTIME $\neq$ NPTIME the status of FPT $\neq W[1]$ is not known; but FPT $\neq W[1]$ would imply PTIME $\neq$ NPTIME.

As for MSO, also for FO we have a characterization of the classes with fixed-parameter tractable model-checking problem.

Theorem 5.10 ([21]) *Assume* FPT $\neq W[1]$. *Let* C *be a class of graphs closed under minors. Then,*

$$\text{P-MC}(\text{C}, \text{FO}) \in \text{FPT} \quad \textit{iff} \quad \text{C} \textit{ is not the class of all graphs.}$$

6 The model-checking problem for tree-like sentences

Stockmeyer [29] showed that the satisfiability problem for quantified boolean formulas is PSPACE-complete, thus:

Theorem 6.1 MC(STR, FO) *is* PSPACE-*complete.*

To get feasible instances of the model-checking problem, so far we restricted the class of structures (to classes of bounded tree-width) but considered the class of all sentences of a given logic. Now we shall restrict the class of sentences.

A *literal* is an atomic or a negated atomic formula. A first-order formula is *existential* if it is built from literals with the connectives $\wedge, \vee$ and with the existential quantifier. Every existential formula is equivalent to a Σ_1-*formula*, i.e., to a formula of the form $\exists x_1 \ldots \exists x_k \psi$ with quantifier-free ψ. And every Σ_1-formula is equivalent to a disjunction of conjunctive formulas, where a *conjunctive formula* (or, *conjunctive query*) is a formula of the form $\exists y_1 \ldots \exists y_m (\lambda_1 \wedge \ldots \wedge \lambda_l)$ with literals λ_i. Denote by Σ_1 and CONJ the class of Σ_1-formulas and the class of conjunctive formulas, respectively.

In a certain sense, CONJ is the most simple "classical" subclass of FO containing sentences. Nevertheless, concerning the model-checking problem for CONJ, the results are disappointing. Since $\exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i < j \leq k} E x_i x_j$ is a conjunctive sentence expressing that there is a clique of size k, one gets:

Theorem 6.2 ([5]) $\mathrm{MC}(\mathrm{STR}, \mathrm{CONJ})$ *is* NPTIME-*complete.*

Theorem 6.3 ([10]) *If* $\mathrm{FPT} \neq W[1]$, *then* $\text{P-MC}(\mathrm{STR}, \mathrm{CONJ}) \in W[1] \setminus \mathrm{FPT}$ (*in fact,* $\text{P-MC}(\mathrm{STR}, \mathrm{CONJ})$ *is* $W[1]$-*complete under parameterized reductions*).

Let us have a look at the canonical algorithm deciding whether a first-order sentence φ holds in $\mathcal{A}$: Starting with the atomic subformulas of φ this algorithm inductively computes the set $\psi(\mathcal{A})$ for every subformula $\psi(\bar{x})$ of φ, therefore the algorithm can take up to $O(\|\varphi\| \cdot |A|^{|\mathrm{var}(\varphi)|})$ number of steps. This motivates the following restriction: We consider the *finite variable fragments* FO^s of FO. For $s \geq 1$ denote by FO^s the class of formulas of FO that contain at most s variables; define CONJ^s, $\Sigma^s, \ldots$ similarly.

For a class C of structures and a set L of first-order formulas we denote by $\mathrm{EVAL}(\mathrm{C}, \mathrm{L})$ the *evaluation problem* for C and L:

Input:	Structure $\mathcal{A} \in \mathrm{C}$; formula φ in L.
Problem:	Compute $\varphi(\mathcal{A})$.

The canonical algorithm mentioned above shows:

Theorem 6.4 ([31]) *For every* $s \geq 1$, $\mathrm{EVAL}(\mathrm{STR}, \mathrm{FO}^s) \in \mathrm{PTIME}$ *and thus,* $\mathrm{MC}(\mathrm{STR}, \mathrm{FO}^s) \in \mathrm{PTIME}$.

6.1 Formulas of bounded tree-width

What does FO^s have to do with tree-decompositions, the theme of our article? We explain the relationship and derive further results.

For a first-order formula φ let $\mathrm{at}(\varphi)$ be the set of atomic subformulas of φ, and let $\mathrm{free}(\varphi)$ and $\mathrm{var}(\varphi)$ be the set of free and the set of all variables of φ, respectively.

The *graph* $\mathcal{G}(\varphi)$ of φ has vertex set $G(\varphi) := \mathrm{var}(\varphi)$ and edge set

$$\begin{aligned} E^{\mathcal{G}(\varphi)} \;&:=\; \{(x, y) \mid x \neq y,\ x, y \in \mathrm{var}(\alpha) \text{ for some } \alpha \in \mathrm{at}(\varphi)\} \\ &\;\cup\; \{(x, y) \mid x \neq y,\, x, y \in \mathrm{free}(\varphi)\}, \end{aligned}$$

i.e., $E^{\mathcal{G}(\varphi)}$ takes care of the "input relations" (the atomic relations) and the "output relation" $\varphi(\_)$. By definition, the *tree-width* of φ is the tree-width of the graph $\mathcal{G}(\varphi)$.

For example, in graphs the sentence $\varphi := \exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i < j \leq k} E x_i x_j$ expresses that there is a clique of size k and $\mathcal{G}(\varphi) = \mathcal{K}_k$ is the k-clique with vertex set

$\{x_1, \ldots, x_k\}$. Similarly, $\varphi := \exists x_1 \ldots \exists x_k (\bigwedge_{1 \leq i < k} Ex_i x_{i+1} \wedge Ex_k x_1)$ expresses that there is a cycle of size k and $\mathcal{G}(\varphi) = \mathcal{C}_k$ is such a cycle.

If L is a fragment of FO and C a class of graphs, we denote by L(C) the set of L-formulas φ with $\mathcal{G}(\varphi) \in$ C. Thus, for $w \geq 1$, L(STR_w) denotes the set of L-formulas of tree-width $\leq w$.

We state a first relationship between the finite variable fragments and the fragments of bounded tree-width. Note that $\mathrm{FO}^{w+1} \subseteq \mathrm{FO}(\mathrm{STR}_w)$. In fact, for $\varphi \in \mathrm{FO}^{w+1}$ with $\mathrm{var}(\varphi) = \{x_1, \ldots, x_{w+1}\}$ we have the tree-decomposition consisting of the single block $\{x_1, \ldots, x_{w+1}\}$. Moreover:

Proposition 6.5 ([24, 14]) *Let $w \geq 1$. There is a linear time algorithm that, applied to a formula in* $\mathrm{CONJ}(\mathrm{STR}_w)$*, yields an equivalent existential formula in* FO^{w+1}*.*

We sketch a proof: Let $\varphi(\bar{x}) = \exists y_1 \ldots \exists y_m (\lambda_1 \wedge \ldots \wedge \lambda_l)$ be of tree-width $\leq w$. We compute a tree-decomposition $(\mathcal{T}, (X_t)_{t \in T})$ of $\mathcal{G}(\varphi)$ of tree-width $\leq w$ (cf. 5.2). Since the variables in $\bar{x}$ form a clique in $\mathcal{G}(\varphi)$ there must be a single block containing all these variables; we choose a corresponding node of the tree as root $r^{\mathcal{T}}$. For $t \in T$ let λ_t be the conjunction of the literals λ_i such that $\mathrm{var}(\lambda_i) \subseteq X_t$. By induction from the leaves to the root $r^{\mathcal{T}}$ we define, for $t \in T$ with children $t_1, \ldots, t_n$ and parent s, the formula φ_t by

$$\varphi_t \quad := \quad \exists y_1 \ldots \exists y_k (\lambda_t \wedge \varphi_{t_1} \wedge \ldots \wedge \varphi_{t_n}),$$

where $X_t \setminus X_s = \{y_1, \ldots, y_k\}$ (and $X_{(r^{\mathcal{T}})} \setminus \{\bar{x}\} = \{y_1, \ldots, y_k\}$). Then, φ and $\varphi_{(r^{\mathcal{T}})}$ are equivalent; since every subformula of $\varphi_{(r^{\mathcal{T}})}$ has at most $w+1$ free variables, $\varphi_{(r^{\mathcal{T}})}$ (and hence, φ) is equivalent to a formula in FO^{w+1}. $\Box$

Proposition 6.5 and Theorem 6.4 yield:

Corollary 6.6 ([7]) *Let $w \geq 1$. Then,* $\mathrm{EVAL}(\mathrm{STR}, \mathrm{CONJ}(\mathrm{STR}_w)) \in \mathrm{PTIME}$ *and thus,* $\mathrm{MC}(\mathrm{STR}, \mathrm{CONJ}(\mathrm{STR}_w)) \in \mathrm{PTIME}$*.*

For later purposes we sketch a direct proof of the last claim of the Corollary, namely of $\mathrm{MC}(\mathrm{STR}, \mathrm{CONJ}(\mathrm{STR}_w)) \in \mathrm{PTIME}$. Let $\varphi = \exists y_1 \ldots \exists y_m (\lambda_1 \wedge \ldots \wedge \lambda_l) \in \mathrm{CONJ}(\mathrm{STR}_w)$ be a sentence. Let $(\mathcal{T}, (X_t)_{t \in T})$ and λ_t be as in the preceding proof.

Assume given a structure $\mathcal{A}$. From the leaves to the root $r^{\mathcal{T}}$ we compute, for every node t with children $t_1, \ldots, t_n$, the set Γ_t of assignments $\gamma : X_t \to A$ that satisfy λ_t in $\mathcal{A}$ and that, for $i = 1, \ldots, n$, are compatible with at least one $\gamma' \in \Gamma_{t_i}$ (γ and γ' are *compatible* if they agree on $X_t \cap X_{t_i}$). One can verify that $\mathcal{A} \models \varphi$ iff $\Gamma_{(r^{\mathcal{T}})} \neq \emptyset$. As $|\Gamma_t| \leq |A|^{w+1}$ we obtain the claim. $\Box$

Since a Σ_1-sentence can effectively be transformed into a disjunction of conjunctive formulas[6], we get:

[6]It is unlikely that such a transformation can be done in polynomial time.

Corollary 6.7 ([14]) *Let $w \geq 1$. Then,* $\text{P-MC}(\text{STR}, \Sigma_1(\text{STR}_w)) \in \text{FPT}$.

Quite recently it has been shown that for conjunctive formulas these results are the best possible ones. Again the notion of tree-decomposition allows to characterize the feasible instances of the model-checking problem.

Theorem 6.8 ([22]) *Let* C *be a class of graphs. Then*

1. *Assume* PTIME $\neq$ NPTIME. *If* C *is closed under minors, then*

$$\text{MC}(\text{STR}, \text{CONJ}(\text{C})) \in \text{PTIME} \quad \textit{iff} \quad \text{tw}(\text{C}) < \infty.$$

2. *Assume* FPT $\neq W[1]$. *Then,*

$$\text{P-MC}(\text{STR}, \text{CONJ}(\text{C})) \in \text{FPT} \quad \textit{iff} \quad \text{tw}(\text{C}) < \infty.$$

We give an application of 6.7. Let $\mathcal{A}$ be a τ-structure, $|A| = n$, and $A = \{a_1, \ldots, a_n\}$. Set

$$\varphi_{\mathcal{A}} := \exists x_1 \ldots \exists x_n \bigwedge_{R \in \tau,\, (a_{i_1}, \ldots, a_{i_m}) \in R^{\mathcal{A}}} R x_{i_1} \ldots x_{i_m}.$$

Then, for every τ-structure $\mathcal{B}$,

$$\mathcal{B} \models \varphi_{\mathcal{A}} \quad \text{iff} \quad \text{there is a homomorphism from } \mathcal{A} \text{ into } \mathcal{B}.$$

Since $\mathcal{G}(\mathcal{A}) \cong \mathcal{G}(\varphi_{\mathcal{A}})$ we have by 6.6:

Corollary 6.9 ([26]) *Let $w \geq 1$. There is a* PTIME*-algorithm solving*

Input:	*Structure $\mathcal{A}$ of tree-width $\leq w$; structure $\mathcal{B}$.*
Problem:	*Is there a homomorphism from $\mathcal{A}$ into $\mathcal{B}$?*

Recalling that the cycles $\mathcal{C}_k$ have tree-width 2, we see that there is a PTIME-algorithm that, for every cycle $\mathcal{C}_k$ and every graph $\mathcal{G}$, decides whether $\mathcal{G}$ contains a homomorphic image of $\mathcal{C}_k$. But what if we want to know whether $\mathcal{G}$ contains a k-cycle? The following proposition shows to what extend we can generalize the results from homomorphisms to embeddings (= injective homomorphisms). A sentence expressing the existence of an embedding of $\mathcal{A}$ is

$$\psi_{\mathcal{A}} := \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} \neg x_i = x_j \wedge \bigwedge_{R \in \tau,\, (a_{i_1}, \ldots, a_{i_m}) \in R^{\mathcal{A}}} R x_{i_1} \ldots x_{i_m}).$$

For a first-order formula φ let $\varphi^{\neq}$ be the formula obtained from φ by deleting all subformulas $x = y$ which occur negatively (i.e., in the scope of an odd number of negation symbols) in φ. Thus, $\psi_{\mathcal{A}}^{\neq} = \varphi_{\mathcal{A}}$. Moreover, let $\Sigma_1^{\neq}(\text{STR}_w)$ be the class of Σ_1-formulas φ such that $\varphi^{\neq}$ has tree-width $\leq w$. Clearly, $\Sigma_1(\text{STR}_w) \subseteq \Sigma_1^{\neq}(\text{STR}_w)$. The following extension of 6.7 holds:

Proposition 6.10 ([14]) *Let $w \geq 1$. Then,* P-MC(STR, $\Sigma_1^{\neq}(\mathrm{STR}_w)$) $\in$ FPT.

The previous results cannot be extended literally to the class of all existential first-order formulas: For every $k \geq 3$ the existential sentence

$$\varphi_k := \exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i < j \leq k} \exists y \exists z (y = x_i \wedge z = x_j \wedge Eyz)$$

saying that a graph contains a k-clique has tree-width 2. It is well-known that no FO^s contains all the φ_k's and that there is no PTIME-algorithm checking, for all graphs $\mathcal{G}$ and all $k \geq 1$, whether $\mathcal{G}$ satisfies φ_k (unless PTIME=NPTIME).

To generalize our results for conjunctive formulas we proceed as follows. First note that a conjunctive formula

$$\varphi(\bar{x}) = \exists y_1 \ldots \exists y_m (\lambda_1 \wedge \ldots \wedge \lambda_l)$$

corresponds to the non-recursive datalog rule

$$Q\bar{x} \ \leftarrow \ \lambda_1, \ldots, \lambda_l$$

where Q is a new relation symbol. The rule is *non-recursive*, since Q – the *head symbol* – does not occur in the body of the rule. Conversely, a non-recursive datalog rule

$$Q\bar{x} \ \leftarrow \ \lambda_1, \ldots, \lambda_l,$$

where we always tacitly assume that $\{\bar{x}\} \subseteq \mathrm{var}(\lambda_1 \wedge \ldots \wedge \lambda_l)$, is equivalent to the conjunctive formula $\varphi(\bar{x}) := \exists y_1 \ldots \exists y_m (\lambda_1 \wedge \ldots \wedge \lambda_l)$ with $\{\bar{y}\} = \mathrm{var}(\lambda_1 \wedge \ldots \wedge \lambda_l) \setminus \{\bar{x}\}$. A *non-recursive datalog program* is a set of non-recursive datalog rules:

$$\begin{array}{lcl} Q\bar{x} & \leftarrow & \lambda_{11}, \ldots, \lambda_{1l_1} \\ & \vdots & \\ Q\bar{x} & \leftarrow & \lambda_{s1}, \ldots, \lambda_{sl_s}. \end{array}$$

Q is its *head symbol.*

While conjunctive formulas correspond to datalog rules, first-order formulas correspond to NRSD-programs, i.e., to *non-recursive stratified datalog programs.* A NRSD-program Π is a sequence $\Pi := (\Pi_1, \ldots, \Pi_d)$ of non-recursive datalog programs Π_i with the property that for $1 \leq i < j \leq d$ the head symbol of Π_j does not occur in Π_i. The Π_i's are called the *strata.* The value $\Pi(\mathcal{A})$ of Π in the structure $\mathcal{A}$ is the value of the head symbol of Π_d obtained by evaluating first Π_1, then Π_2, ... , and finally Π_d (e.g., cf. [12] for details). It is well-known that:

Proposition 6.11 *First-order logic and* NRSD*-programs have the same expressive power.*

For example, the sentence $\forall x \exists y(Exy \vee \neg Py)$ is equivalent to the program

$$\begin{array}{lcl}
Q_1 x & \leftarrow & Exy \\
Q_1 x & \leftarrow & \neg Py, x = x \\
\multicolumn{3}{l}{\dotfill} \\
Q_2 x & \leftarrow & \neg Q_1 x \\
\multicolumn{3}{l}{\dotfill} \\
Q_3 & \leftarrow & Q_2 x \\
\multicolumn{3}{l}{\dotfill} \\
Q_4 & \leftarrow & \neg Q_3.
\end{array}$$

By definition, a datalog rule has *tree-width* $\leq w$ if the corresponding conjunctive formula does. And a NRSD-program has *tree-width* $\leq w$ if every rule does. Denote by NRSD(w) the set of NRSD programs of tree-width $\leq w$. By using 6.6 one gets:

Corollary 6.12 ([13]) *Let* $w \geq 1$. *Then,* EVAL(STR, NRSD(w)) $\in$ PTIME.

We mentioned that NRSD has the same expressive power as FO. But what fragment of FO corresponds to NRSD(w)?

Theorem 6.13 ([13]) *Let* $w \geq 1$. *For every first-order formula* φ *we have*

$$\varphi \underset{\sim}{\in} \mathrm{NRSD}(w) \quad \textit{iff} \quad \varphi \underset{\sim}{\in} \mathrm{FO}^{w+1}$$

($\varphi \underset{\sim}{\in}$ L *means that* φ *is equivalent to a formula or program in* L). *More precisely: There is a linear time algorithm associating with every formula in* FO^{w+1} *an equivalent program in* NRSD(w). *And there is an algorithm associating with every* NRSD(w)*-program an equivalent* FO^{w+1}*-formula.*

Using the first algorithm we see that the previous corollary implies Theorem 6.4. By contrast, the canonical algorithm translating NRSD-programs into first-order formulas is not a linear time algorithm: NRSD-programs allow more succinct representations than formulas, since a single stratum can take care of a subformula occurring several times in a given formula.

So far the evaluation methods used the fact that, in a structure $\mathcal{A}$, a conjunctive formula φ of tree-width $\leq w$ can be evaluated along a corresponding tree-decomposition $(\mathcal{T}, (X_t)_{t \in T})$ of $\mathcal{G}(\varphi)$: at any node t we evaluate the literals of φ whose variables are in X_t; this can be done in time $O(\|\mathcal{A}\|^{w+1})$.

6.2 Acyclic formulas

We turn to Yannakakis' approach: He considered conjunctive formulas φ whose graphs $\mathcal{G}(\varphi)$ have tree-decompositions $(\mathcal{T}, (X_t)_{t \in T})$, where each node $t \in T$ is "guarded" by a positive atomic subformula $R\bar{x}$, i.e., $X_t = \{\bar{x}\}$. Then, when evaluating φ, at the node t one can restrict to the tuples in $R^{\mathcal{A}}$. This yields a fast algorithm. Let us give precise definitions and results.

A conjunctive formula $\varphi = \exists y_1 \ldots \exists y_m(\lambda_1 \wedge \ldots \wedge \lambda_l)$ is *acyclic* if there is a tree-decomposition $(\mathcal{T}, (X_t)_{t\in T})$ of $\mathcal{G}(\varphi)$ such that for every $t \in T$ there is a positive λ_i s.t. $X_t = \mathrm{var}(\lambda_i)$. Then, λ_i is called a *guard* at t.

For example, for $n \geq 3$, the sentence $\varphi := \exists x_1 \ldots \exists x_n \bigwedge_{1\leq i<n} Ex_ix_{i+1} \wedge Ex_nx_1$ is not acyclic, since every tree-decomposition of φ has at least one node with ≥ 3 variables. However, $\exists x_1 \ldots \exists x_n \bigwedge_{1\leq i<n} Ex_ix_{i+1}$ is acyclic.

We denote by ACYCLIC the class of acyclic conjunctive formulas. Yannakakis' result reads as follows:

Theorem 6.14 (([32]) $\mathrm{EVAL}(\mathrm{STR}, \mathrm{ACYCLIC}) \in \mathrm{PTIME}$ *and therefore,* $\mathrm{MC}(\mathrm{STR}, \mathrm{ACYCLIC}) \in \mathrm{PTIME}$.

Again, we can generalize this result via datalog. We call a datalog rule *acyclic*, if the corresponding conjunctive formula is acyclic. An NRSD-program is *acyclic*, if every rule is acyclic and, moreover, has a tree-decomposition whose guards can be chosen among the literals containing no head symbols of previous strata. Denote by ACY-NRSD the class of acyclic NRSD-programs.

Then, in a straightforward manner one obtains the following generalization of Yannakakis' result:

Theorem 6.15 ([13]) $\mathrm{EVAL}(\mathrm{STR}, \text{ACY-NRSD}) \in \mathrm{PTIME}$ *and therefore,* $\mathrm{MC}(\mathrm{STR}, \text{ACY-NRSD}) \in \mathrm{PTIME}$.

Again we ask: What fragment of first-order logic corresponds to the class of acyclic NRSD-programs? It turns out that this is just the guarded fragment, which has recently been studied intensively ([2, 17]). We recall its definition: The set GF of formulas of the *guarded fragment* is obtained by closing the set of atomic first-order formulas under the connectives $\wedge$, $\vee$, and $\neg$ and under the rule

> If φ is a GF-formula, α is atomic, and $\mathrm{free}(\varphi) \subseteq \mathrm{var}(\alpha)$ then, for every tuple $\bar{y}$ of variables, $\exists\bar{y}(\alpha \wedge \varphi)$ is a GF-formula.

As indicated above we have:

Theorem 6.16 ([13]) *For every first-order sentence φ we have*

$$\varphi \underset{\sim}{\in} \text{ACY-NRSD} \quad \textit{iff} \quad \varphi \underset{\sim}{\in} \mathrm{GF}.$$

Moreover, there is a quadratic time algorithm that associates with every GF-sentence an equivalent ACY-NRSD-program. Hence:

Theorem 6.17 ([17]) $\mathrm{MC}(\mathrm{STR}, \mathrm{GF}) \in \mathrm{PTIME}$.

Gottlob, Leone, and Scarcello [18] introduce the notion of bounded hypertree-width, a further generalization of (positive) acyclic conjunctive formulas; in fact, acyclic conjunctive queries are conjunctive queries of hypertree-width one. They show that the model-checking problem for most classes of tree-like conjunctive queries considered so far is in the parallel complexity class LOGCFL and actually is complete for this class.

References

[1] M. Ajtai and Y. Gurevich. Datalog vs. first-order logic. *Journal of Computer and System Sciences*, 49:562–588, 1994.

[2] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of first-order logic, 1996. ILLC Research Report ML-96-03, University of Amsterdam.

[3] H.L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–126, 1988.

[4] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[5] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[6] V. Chvátal On the computational complexity of finding a kernel. *Centre des Recherches Mathématiques, Université de Montréal*, Report No. CRM–300, 1973.

[7] Ch. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In Ph. Kolaitis and F. Afrati, editors, *Proceedings of the 5th International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 56–70. Springer-Verlag, 1997.

[8] B. Courcelle. Graphs and monadic second-order logic, some open problems. *Bulletin of the EATCS*, 49:110–124, 1993.

[9] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194–242. Elsevier Science Publishers, 1990.

[10] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness II. *Theoretical Computer Science*, 141:109–131, 1995.

[11] R.G. Downey and M.R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.

[12] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.

[13] J. Flum, M. Frick and M. Grohe. Query evaluation via tree-decompositions. In J. van den Busche and V. Vianu, editors, *Proceedings of the 8th International Conference on Database Theory*, Lecture Notes in Computer Science, Springer-Verlag, 2001. To appear.

[14] J. Flum and M. Grohe. Fixed-parameter tractability and logic. To appear in *SIAM Journal of Computing.*

[15] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable graphs. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 1644. Springer-Verlag, 1999.

[16] H. Gaifman. On local and non-local properties.. In *Proceedings of the Herbrand Symposium, Logic Colloquium' 81,* 105–135, North Holland,1982.

[17] G. Gottlob, E. Grädel, and H. Veith. Datalog lite: Temporal versus deductive reasoning in verification. Technical Report DBAI-TR-98-22, Technische Universität Wien, 1998.

[18] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the 18th ACM Symposium on Principles of Database Systems,* 21–32, 1999.

[19] M. Grohe. Local tree-width, excluded minors, and approximation algorithms. Submitted.

[20] M. Grohe. Computing crossing numbers in quadratic time. In preparation.

[21] M. Grohe. Descriptive and parameterized complexity. In *Computer Science Logic 1999*, Lecture Notes in Computer Science 1683, Springer-Verlag, 1999.

[22] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? Submitted.

[23] R. Halin. S-functions for graphs. *Journal of Geometry*, 8:171–186, 1976.

[24] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, pages 205–213, 1998.

[25] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[26] J. Plehn and B. Voigt. Finding minimally weighted subgraphs. In R. Möhring, editor, *Graph-Theoretic Concepts in Computer Science, WG '90*, volume 484 of *Lecture Notes in Computer Science*, pages 18–29. Springer-Verlag, 1990.

[27] N. Robertson and P.D. Seymour. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[28] N. Robertson and P.D. Seymour. Graph minors V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[29] L.J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory.* PhD thesis, Department of Electrical Engineering, MIT, 1974.

[30] J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order arithmetic.. *Mathematical Systems Theory*, 2:57–81, 1968.

[31] M. Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, pages 266–276, 1995.

[32] M. Yannakakis. Algorithms for acyclic database schemes. In *7th International Conference on Very Large Data Bases*, pages 82–94, 1981.