The Tractability Frontier of Graph-Like First-Order Query Sets

Hubie Chen
Departamento LSI, Universidad del País Vasco
and IKERBASQUE, Basque Foundation for Science
E-20018 San Sebastián, Spain
hubie.chen@ehu.es

Abstract

We study first-order model checking, by which we refer to the problem of deciding whether or not a given first-order sentence is satisfied by a given finite structure. In particular, we aim to understand on which sets of sentences this problem is tractable, in the sense of parameterized complexity theory. To this end, we define the notion of a graph-like sentence set, which definition is inspired by previous work on first-order model checking wherein the permitted connectives and quantifiers were restricted. Our main theorem is the complete tractability classification of such graph-like sentence sets, which is (to our knowledge) the first complexity classification theorem concerning a class of sentences that has no restriction on the connectives and quantifiers. To present and prove our classification, we introduce and develop a novel complexity-theoretic framework which is built on parameterized complexity and includes new notions of reduction.

Categories and Subject Descriptors F.4.1 [*Mathematical logic and formal languages*]: Mathematical logic—Computational logic; H.2.3 [*Database management*]: Languages—Query languages

General Terms Theory

Keywords first-order queries, query evaluation, parameterized complexity, treewidth

1. Introduction

Model checking, the problem of deciding if a logical sentence holds on a structure, is a fundamental computational task that appears in many guises throughout computer science. In this article, we study *first-order model checking*, by which we refer to the case of this problem where one wishes to evaluate a first-order sentence on a finite structure. This case is of principal interest in database theory, where first-order sentences form a basic, heavily studied class of *database queries*, and where it is well-recognized that the problem of evaluating such a query on a database can be taken as a formulation of first-order model checking. Indeed, the investigation of model checking in first-order logic entails an examination of one of the simplest, most basic logics, and it can be expected that understanding of the first-order case should provide a well-founded basis for studying model checking in other logics, such as those typically considered in verification and database theory. First-order model checking is well-known to be intractable in general: it is PSPACE-complete.

As has been articulated [11, 16], the typical model-checking situation in the database and verification settings is the evaluation of a relatively short sentence on a relatively large structure. Consequently, it has been argued that, in measuring the time complexity of model checking, one could reasonably allow a slow (non-polynomial-time) preprocessing of the sentence, so long as the desired evaluation can be performed in polynomial time following the preprocessing. Relaxing polynomial-time computation to allow arbitrary preprocessing of a *parameter* of a problem instance yields, in essence, the notion of *fixed-parameter tractability*. This notion of tractability is the base of *parameterized complexity theory*, which provides a taxonomy for reasoning about and classifying problems where each instance has an associated parameter. We utilize this paradigm, and focus the discussion on this form of tractability (here, the sentence is the parameter).

A typical way to understand which types of sentences are well-behaved and exhibit desirable, tractable behavior is to simply consider model checking relative to a set Φ of sentences, and to attempt to understand on which sets one has tractable model checking. We restrict attention to sets of sentences having bounded arity.[1] Here, there have been successes in understanding which sets of sentences are tractable (and which are not) in fragments of first-order logic described by restricting the connectives and quantifiers that may be used: there are systematic classification results for so-called conjunctive queries [8, 12, 13] (formed using the connectives and quantifiers in $\{\wedge, \exists\}$) existential positive queries [4] ($\{\wedge, \vee, \exists\}$), and quantified conjunctive queries [5, 7] ($\{\wedge, \exists, \forall\}$). However, to the best of our knowledge, there has been no classification theorem for general first-order logic, without any restriction on the connectives and quantifiers. In this article, we present the first such classification.

Our approach. In the fragments of first-order logic where the only connective permitted is one of the binary connectives ($\{\wedge, \vee\}$)—such as those of conjunctive queries and quantified conjunctive queries—a heavily studied approach to describing sets Φ of sentences is a *graphical approach*. In this graphical approach, one studies a sentence set Φ if it is *graphical* in the following sense: if one prenex sentence ϕ is contained in Φ and a second prenex sentence ψ has the same prefix as ϕ and also has the same graph as ϕ, then ψ is also in Φ. (By the graph of a prenex sentence ϕ, we mean the graph whose vertices are the variables of ϕ and where two vertices are adjacent if they occur together in an atomic formula.) In the fragments where it was considered, this approach of studying graphical sentence sets is not only a natural way to coarsen the project of classifying all sets Φ of sentences, but in fact, can be used cleanly as a key module in obtaining general classifications of sentence sets: such general classifications have recently been proved by using the respective graphical classifications as black boxes [7, 8].

[1] Note that in the case of unbounded arity, complexity may depend on the choice of representation of relations [6].

In this article, we adapt this graphical approach to the full first-order setting. To explain how this is done, consider that a graphical set Φ of sentences satisfies certain syntactic closure properties. For instance, if one takes a sentence from such a graphical set Φ and replaces the relation symbol of an atomic formula, the resulting sentence will have the same graph, and will hence continue to be in Φ; we refer to this property of Φ as *replacement closure*. As another example, if one rewrites a sentence in Φ by invoking associativity or commutativity of the connective $\wedge$, the resulting sentence will likewise still have the same graph and will hence be contained in Φ also. Inspired by these observations, we define a sentence set Φ to be *graph-like* if it is replacement closed and also closed under certain well-known syntactic transformations, such as associativity and commutativity of the binary connectives (see Section 2 for the full definition).

Our principal result is the complete tractability characterization of graph-like sentence sets (see Theorem 6.2 and the corollaries that follow). In particular, we introduce a measure on first-order formulas which we call *thickness*, and show that a set of graph-like sentences is tractable if and only if it has bounded thickness (under standard complexity-theoretic assumptions). In studying unrestricted first-order logic, we believe that our building on the syntactic, graphical approach—which has an established, fruitful tradition—will facilitate the formulation and obtention of future, more general results.

As evidence of our result's generality and of its faithfulness to the graphical approach, we note that the graphical classification of quantified conjunctive queries [5] can be readily derived from our main classification result (this will be discussed in the full version of this paper); it follows readily that the dual graphical classification of quantified disjunctive queries (also previously derived [5]) can be dually derived from our main classification result. We therefore give a single classification theorem that naturally unifies together these two previous classifications. Indeed, we believe that the technology that we introduce to derive our classification yields a cleaner, deeper and more general understanding of these previous classifications. Observe that, for each of those classifications, since only one binary connective is present, the two quantifiers behave asymmetrically; this is in contrast to the present situation, where in building formulas, wherever a formula may be constructed, its dual may be as well.

Parameterized complexity. An increasing literature investigates the following general situation:

> Given a parameterized problem P
> whose instances consist of two parts, and a set S,
> define $P[\![S]\!]$ to be the restricted version of P where
> (x, y) is admitted as an instance iff $x \in S$.
>
> Then, attempt to classify and understand, over all sets S,
> the complexity of the problem $P[\![S]\!]$.

Examples of such classifications and studies include [5, 6, 8–10, 12–14]. It is our view that this literature suffers from the defect that there is no complexity-theoretic framework for discussing the families of problems obtained thusly. As a consequence, different authors and different articles used divergent language and notions to present hardness results on and reductions between such problems, and applied different computability assumptions on the sets S considered. We attempt to make a foundational contribution and to ameliorate this state of affairs by presenting a complexity-theoretic framework for handling and classifying problems of the described form. In particular, we introduce notions such as reductions and complexity classes for problems of the above type, which we formalize as *case problems* (Section 4). Although we do not carry out this exercise here, we believe that most of the results in the mentioned literature can be shown to be naturally and transparently expressible within our framework.

In order to derive our classification, we present (within our complexity-theoretic framework) a new notion of reduction which we call *accordion reduction* and which is crucial for the proof of our hardness result. (See Sections 7 and 8 for further discussion.) We believe that this notion of reduction may play a basic role in future classification projects of the form undertaken here.

Let us emphasize that, while the establishment of our main classification theorem makes use of the complexity-theoretic framework and accompanying machinery that was just discussed (and is presented in Sections 4 and 7), this framework and machinery is fully generic in that it does not make any reference to and is not specialized to the model checking problem. We believe and hope that the future will find this framework to be a suitable basis for presenting, developing and discussing complexity classification results.

2. Preliminaries

When $g : A \to B$ and $h : B \to C$ are mappings, we will typically use $h(g)$ to denote their composition. When f is a partial mapping, we use $\mathsf{dom}(f)$ to denote its domain, and we use $f \restriction S$ to denote its restriction to the set S. We will use π_i to denote the ith projection, that is, the mapping that, given a tuple, returns the value in the tuple's ith coordinate. For a natural number k, we use $\underline{k}$ to denote the set $\{1, \ldots, k\}$.

2.1 First-order logic

We use the syntax and semantics of first-order logic as given by a standard treatment of the subject. In this article, we restrict to relational first-order logic, so the only symbols in signatures are relation symbols. We use letters such as $\mathbf{A}$, $\mathbf{B}$ to denote structures and A, B to denote their respective universes. The reader may assume for concreteness that relations of structures are represented using lists of tuples, although in general, we will deal with the setting of bounded arity, and natural representations of relations will be (for the complexity questions at hand) equivalent to this one. We assume that equality is not built-in to first-order logic, so a *formula* is created from *atoms*, the usual connectives ($\neg$, $\wedge$, $\vee$) and quantification ($\exists$, $\forall$); by an *atom*, we mean a formula $R(v_1, \ldots, v_k)$ where a relation symbol is applied to a tuple of variables (of the arity of the symbol). A formula is *positive* if it does not contain negation ($\neg$). We use $\mathsf{free}(\phi)$ to denote the set of free variables of a formula ϕ. The *width* of a formula ϕ, denoted by $\mathsf{width}(\phi)$, is defined as the maximum of $|\mathsf{free}(\psi)|$ over all subformulas ψ of ϕ. The *arity* of a formula is the maximum arity over all relation symbols that occur in the formula.

We use $\phi_1 \wedge \cdots \wedge \phi_n$ as notation for $(\cdots((\phi_1 \wedge \phi_2) \wedge \phi_3) \cdots)$, and $\phi_1 \vee \cdots \vee \phi_n$ is defined dually. We refer to a formula of the shape $\phi_1 \wedge \cdots \wedge \phi_n$ as a *conjunction*, and respectively refer to a formula of the shape $\phi_1 \vee \cdots \vee \phi_n$ as a *disjunction*. Let Φ be a set of formulas. A *positive combination* of formulas from Φ is a formula in the closure of Φ under conjunction and disjunction. A *CNF* of formulas from Φ is a conjunction of disjunctions of formulas from Φ, and a *DNF* of formulas from Φ is a disjunction of conjunctions of formulas from Φ.

We will use the following terminology which is particular to this article. A subformula ψ of a formula ϕ is a *positively combined subformula* if, in viewing ϕ as a tree, all nodes on the unique path from the root of ϕ to the parent of the root of ψ (inclusive) are conjunctions or disjunctions. We say that a formula ϕ is *variable-loose* if no variable is quantified twice, and no variable is both quantified and a free variable of ϕ. We say that a formula is *symbol-loose* if no relation symbol appears more than once in the formula.

We say that a formula is *loose* if it is both variable-loose and symbol-loose.

We now present a number of syntactic transformations; that each preserves logical equivalence is well-known.[2]

(α) Associativity and commutativity of $\wedge$ and $\vee$

(β) $\exists x(\bigvee_{i=1}^{n} \phi_i) \equiv \bigvee_{i=1}^{n}(\exists x \phi_i)$,
$\forall y(\bigwedge_{i=1}^{n} \phi_i) \equiv \bigwedge_{i=1}^{n}(\forall y \phi_i)$

(γ) $\exists x(\phi \wedge \psi) \equiv (\exists x \phi) \wedge \psi$ if $x \notin \mathsf{free}(\psi)$,
$\forall y(\phi \vee \psi) \equiv (\forall y \phi) \vee \psi$ if $y \notin \mathsf{free}(\psi)$

(δ) (Distributivity for $\wedge$ and $\vee$)
$\phi \wedge (\psi \vee \psi') \equiv (\phi \wedge \psi) \vee (\phi \wedge \psi')$,
$\phi \vee (\psi \wedge \psi') \equiv (\phi \vee \psi) \wedge (\phi \vee \psi')$

(ϵ) (DeMorgan's laws)
$\neg\exists v \phi \equiv \forall v \neg\phi$, $\neg\forall v \phi \equiv \exists v \neg\phi$
$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$, $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

We say that a set Φ of formulas is *syntactically closed* if, for each $\phi \in \Phi$, when a formula ϕ' can be obtained from ϕ by applying one of the syntactic transformations (α), (β), (γ), (δ), (ϵ) to a subformula of ϕ, it holds that $\phi' \in \Phi$. The *syntactic closure* of a formula ϕ is the intersection of all syntactically closed sets that contain ϕ.

Let us say that a formula ϕ' on signature σ' is obtainable from a formula ϕ on signature σ by *replacement* if ϕ' can be obtained from ϕ by replacing instances of relation symbols in ϕ with instances of relation symbols from σ' (without making any other changes to ϕ).

Example 2.1. Let ϕ be the formula $\forall y \exists x \exists x'(E(y,x) \wedge E(x,x'))$. Let τ be the signature $\{E, F\}$ where E and F are relation symbols of binary arity. Each of the four formulas ϕ, $\forall y \exists x \exists x'(F(y,x) \wedge E(x,x'))$, $\forall y \exists x \exists x'(E(y,x) \wedge F(x,x'))$, and $\forall y \exists x \exists x'(F(y,x) \wedge F(x,x'))$ is obtainable from ϕ by replacement; moreover, these are the only four formulas over signature τ that are obtainable from ϕ by replacement.

Let us say that a set of formulas Φ is *replacement closed* if, for each $\phi \in \Phi$, when ϕ' is obtainable from ϕ by replacement, it holds that $\phi' \in \Phi$.

Definition 2.2. A set of formulas Φ is *graph-like* if it is syntactically closed and replacement closed.

2.2 Graphs and hypergraphs

When S is a set, we use $K(S)$ to denote the set containing all size 2 subsets of S, that is, $K(S) = \{\{s, s'\} \mid s, s' \in S, s \neq s'\}$. For us, a *graph* is a pair (V, E) where V is a set and $E \subseteq K(V)$.

Here, a *hypergraph* H is a pair $(V(H), E(H))$ consisting of a vertex set $V(H)$ and an edge set $E(H)$ which is a subset of the power set $\wp(V(H))$. We will sometimes specify a hypergraph just by specifying the edge set E, in which case the vertex set is understood to be $\bigcup_{e \in E} e$. We associate a hypergraph $(V(H), E(H))$ with the graph $(V(H), \bigcup_{e \in E(H)} K(e))$, and thereby refer to (for example) the treewidth of or an elimination ordering of a hypergraph.

An *elimination ordering* of a graph (V, E) is a pair

$$((v_1, \ldots, v_n), E')$$

that consists of a superset E' of E and an ordering $v_1, \ldots, v_n$ of the elements of V such that the following property holds: for each vertex v_k, any two distinct lower neighbors v, v' of v_k are adjacent in E', that is, $\{v, v'\} \in E'$; here, a *lower neighbor* of a vertex v_k is a vertex v_i such that $i < k$ and $\{v_i, v_k\} \in E'$. Relative to an elimination ordering e, we define the *lower degree* of a vertex v, denoted by $\mathsf{lower\text{-}deg}(e, v)$, to be the number of lower neighbors that it has; we define $\mathsf{lower\text{-}deg}(e)$ to be the maximum of $\mathsf{lower\text{-}deg}(e, v)$ over all vertices v. We assume basic familiarity with the theory of treewidth [2]. The following is a key property of treewidth that we will utilize; here, we use $\mathsf{tw}(H)$ to denote the treewidth of H.

Proposition 2.3. For each $k \geqslant 2$, there exists a polynomial-time algorithm that, given as input a hypergraph H with a distinguished edge f, will return the following whenever $\mathsf{tw}(H) < k$: an elimination ordering $e = ((v_1, \ldots, v_m), E)$ with $\mathsf{lower\text{-}deg}(e) = \mathsf{tw}(H)$ and where $\{v_1, \ldots, v_{|f|}\} = f$.

The treewidth of (for example) a set of graphs $\mathcal{G}$ is the set $\{\mathsf{tw}(G) \mid G \in \mathcal{G}\}$; it is said to be *unbounded* if this set is infinite, and *bounded* otherwise. We employ similar terminology, in general, when dealing with a complexity measure defined on a class of objects.

3. Parameterized complexity

In this section, we specify the framework of parameterized complexity to be used in this article.

Throughout, we use Σ to denote an alphabet over which languages are defined. As is standard, we will sometimes view elements of $\Sigma^* \times \Sigma^*$ as elements of Σ^*. A *parameterization* is a mapping from Σ^* to Σ^*. A *parameterized problem* is a pair (Q, κ) consisting of a language $Q \subseteq \Sigma^*$ and a parameterization $\kappa : \Sigma^* \to \Sigma^*$. A *parameterized class* is a set of parameterized problems.

Assumption 3.1. We assume that each parameterized problem (Q, κ) has a non-trivial language Q, that is, that neither $Q = \Sigma^*$ nor $Q = \varnothing$.

Remark 3.2. Let us remark on a difference between our setup and that of other treatments. Elsewhere, a *parameterization* is often defined to be a mapping from Σ^* to $\mathbb{N}$, and in the context of query evaluation, the parameterization studied is typically the size of the query. In contrast, this article takes the parameterization to be the query itself. Since there are finitely many queries of any fixed size, model checking on a set of queries will be fixed-parameter tractable (that is, in the class FPT, defined below) under one of these parameterizations if and only if it is under the other. However, we find that—as concerns the theory in this article—taking the query itself to be the parameter allows for a significantly cleaner presentation. One example reason is that the reductions we present will generally be "slice-to-slice", that is, they will send all instances with the same query to instances that share another query. Indeed, to understand the complexity of a set of queries, we will apply a closure operator to pass to a larger set of queries having the same complexity, using the notion of accordion reduction (see Sections 7 and 8); we believe that the theory justifying this passage is most cleanly expressed under the used parameterization. □

We now define what it means for a partial mapping r to be *FPT-computable*; we actually first define a non-uniform version of this notion. This definition coincides with typical definitions in the case that r is a total mapping; in the case that r is a partial mapping, we use a "promise" convention, that is, we do not impose any mandate on the behavior of the respective algorithm in the case that the input x is not in the domain of r.

Definition 3.3. Let $\kappa : \Sigma^* \to \Sigma^*$ be a parameterization.

A partial mapping $r : \Sigma^* \to \Sigma^*$ is *nu-FPT-computable with respect to* κ if there exist a function $f : \Sigma^* \to \mathbb{N}$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for each $k \in \Sigma^*$, there exists an algorithm

[2] Note that in the transformation (γ), we permit that ϕ is not present.

A_k satisfying the following condition: on each string $x \in \mathsf{dom}(r)$ such that $\kappa(x) = k$, the algorithm A_k computes $r(x)$ within time $f(\kappa(x))p(|x|)$.

A partial mapping $r : \Sigma^* \to \Sigma^*$ is *FPT-computable with respect to* κ if there exists a single algorithm A that can play the role of each algorithm A_k in the just-given definition; formally, if there exist a function $f : \Sigma^* \to \mathbb{N}$, a polynomial $p : \mathbb{N} \to \mathbb{N}$, and an algorithm A such that for each $k \in \Sigma^*$, the above condition is satisfied when A_k is set equal to A. □

Definition 3.4. We define FPT to be the class that contains a parameterized problem (Q, κ) if and only if the characteristic function of Q is FPT-computable with respect to κ.

Let us remark that we do not put in effect some of the computability assumptions that are often present in other treatments, for instance, we do not require a computable upper bound on the function f in our definition of *FPT-computable*. This is for simplicity; it can be verified that the theory that we present and carry out can also be done so in the case when such assumptions are present.

We now introduce the notion of a reduction between parameterized problems.

Definition 3.5. Let (Q, κ) and (Q', κ') be parameterized problems. A *FPT-reduction* (respectively, *nu-FPT-reduction*) from (Q, κ) to (Q', κ') is a total mapping g that is FPT-computable (respectively, nu-FPT-computable) with respect to κ such that: (1) for each $x \in \Sigma^*$, it holds that $x \in Q$ if and only if $g(x) \in Q'$; and (2) for each $k \in \Sigma^*$, the set $\kappa'(g(\{x \in \Sigma^* \mid \kappa(x) = k\}))$ is finite.

Assumption 3.6. We assume that each parameterized class C is closed under FPT-reductions, that is, if (Q', κ') is in C and (Q, κ) FPT-reduces to (Q', κ'), then (Q, κ) is in C.

Proposition 3.7. Let (Q, κ), (Q', κ'), and (Q'', κ'') be parameterized problems.

- If g is a FPT-reduction from (Q, κ) to (Q', κ') and h is a FPT-reduction from (Q', κ') to (Q'', κ''), then their composition $h(g)$ is a FPT-reduction from (Q, κ) to (Q'', κ'').
- Similarly, if g is a nu-FPT-reduction from (Q, κ) to (Q', κ') and h is a nu-FPT-reduction from (Q', κ') to (Q'', κ''), then their composition $h(g)$ is a nu-FPT-reduction from (Q, κ) to (Q'', κ'').

Also, each FPT-reduction from (Q, κ) to (Q', κ') is an nu-FPT-reduction from (Q, κ) to (Q', κ').

We may now define, for each parameterized class C, a non-uniform version of the class, denoted by nu-C.

Definition 3.8. (non-uniform classes) When C is a parameterized class, we define nu-C to be the set that contains each parameterized problem that has a nu-FPT-reduction to a problem in C.

Remark 3.9. It is straightforwardly verified that a parameterized problem (Q, κ) is in the class nu-FPT (under the above definitions) if and only if the characteristic function of Q is nu-FPT-computable with respect to κ.

We next present two notions of hardness for parameterized classes.

Definition 3.10. (hardness) Let C be a parameterized class. We say that a problem (Q, κ) is C-hard if every problem in C has a FPT-reduction to (Q, κ). We say that a problem (Q, κ) is non-uniformly C-hard if every problem in C has a nu-FPT-reduction to (Q, κ).

We end this section by observing some basic closure properties of what we call *degree-bounded functions*, which, roughly speaking, are the functions which can serve as the running time of algorithms in the definition of *FPT-computable* (Definition 3.3). Let $\kappa : \Sigma^* \to \Sigma^*$ be a parameterization. A partial function $T : \Sigma^* \to \mathbb{N}$ is *degree-bounded* with respect to κ if there exist a function $f : \Sigma^* \to \mathbb{N}$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that, for each $x \in \mathsf{dom}(T)$, it holds that $T(x) \leqslant f(\kappa(x))p(|x|)$. We will make use of the observation that a partial mapping $h : \Sigma^* \to \Sigma^*$ is FPT-computable with respect to κ if and only if there is a degree-bounded function T (with $\mathsf{dom}(T) \supseteq \mathsf{dom}(h)$) and an algorithm that, for all $x \in \mathsf{dom}(h)$, computes $h(x)$ within time $T(x)$.

The following proposition will be of use in establishing that functions are degree-bounded.

Proposition 3.11. Let κ be a parameterization, and let $T_1, \ldots, T_m : \Sigma^* \to \mathbb{N}$ be partial functions sharing the same domain.

1. If each of $T_1, \ldots, T_m$ is degree-bounded with respect to κ, then $T_1 + \cdots + T_m$ is as well.
2. If each of $T_1, \ldots, T_m$ is degree-bounded with respect to κ, then the product $T_1 \cdots T_m$ is as well.
3. Let $q : \mathbb{N} \to \mathbb{N}$ be a polynomial; if a partial function $T : \Sigma^* \to \mathbb{N}$ is degree-bounded with respect to κ, then $q(T)$ is as well.

4. Case complexity

A number of previous works focus on a decision problem Q where each instance consists of two parts, and obtain restricted versions of the problem by taking sets $S \subseteq \Sigma^*$ and, for each such set, considering the restricted version where one allows only instances where (say) the first of the parts falls into S. This is precisely the type of restriction that we will consider here. We are interested in first-order model checking, which we view as the problem of deciding, given a first-order sentence and a structure, whether or not the sentence holds on the structure; our particular interest is to study restricted versions of this problem where the allowed sentences come from a set S.

It has been useful (see for instance the articles [5, 8]) and is useful in the present article to present reductions between such restricted versions of problems. In order to facilitate our doing this, we present a framework wherein we formalize this type of restricted version of problem as a *case problem*, and then present a notion of reduction for comparing case problems. We believe that our notion of reduction, called *slice reduction*, faithfully abstracts out precisely the key useful properties that are typically present in such reductions in the literature. Note that, in the existing literature, different articles imposed different computability assumptions on the sets S considered (assumptions used include that of computable enumerability, of computability, and of no computability assumption). One feature of our framework is that such reductions can be carried out and discussed independently of whether or not any such computability assumption is placed on the sets S; a general theorem (Theorem 4.5) allows one to derive normal parameterized reductions from slice reductions, where the exact computability of the reduction derivable depends on the computability assumption placed on the sets S.

We now introduce our framework. Suppose that $Q \subseteq \Sigma^* \times \Sigma^*$ is a language of pairs; for a set $T \subseteq \Sigma^*$, we use Q_T to denote the language $Q \cap (T \times \Sigma^*)$ and for a single string $t \in \Sigma^*$, we use Q_t to denote the language $Q \cap (\{t\} \times \Sigma^*)$.

Definition 4.1. A *case problem* consists of a language of pairs $Q \subseteq \Sigma^* \times \Sigma^*$ and a subset $S \subseteq \Sigma^*$, and is denoted $Q[S]$. When $Q[S]$ is a case problem, we use param-$Q[S]$ to denote the parameterized problem (Q_S, π_1).

Ultimately, our purpose in discussing a case problem $Q[S]$ is to understand the complexity of the associated parameterized problem param-$Q[S]$. As mentioned, formalizing the notion of a case problem allows us to cleanly present reductions between such problems.

Remark 4.2. Let (Q, κ) be a parameterized problem. Under the assumption that κ is FPT-computable with respect to itself, the parameterized problem (Q, κ) is straightforwardly verified to be equivalent, under FPT-reduction, to the parameterized problem (Q', π_1) where $Q' = \{(\kappa(x), x) \mid x \in \Sigma^*\}$. Hence, any such given parameterized problem (Q, κ) may be canonically associated to the case problem $Q'[\Sigma^*]$, as one has $\mathsf{param}\text{-}Q'[\Sigma^*] = (Q', \pi_1)$.

We define case-CLIQUE to be the case problem $Q[\Sigma^*]$ where Q contains a pair (k, G) if and only if G is a graph that contains a clique of size k (we assume that both G and k are encoded as strings over Σ); we define case-co-CLIQUE to be the problem $\overline{Q}[\Sigma^*]$.

We now present the notion of *slice reduction*, which allows us to compare case problems.

Definition 4.3. A case problem $Q[S]$ *slice reduces* to a second case problem $Q'[S']$ if there exist:

- a computably enumerable language $U \subseteq \Sigma^* \times \Sigma^*$ and
- a partial function $r : \Sigma^* \times \Sigma^* \times \Sigma^* \to \Sigma^*$ that has $\mathsf{dom}(r) = U \times \Sigma^*$ and is FPT-computable with respect to the parameterization (π_1, π_2)

such that the following conditions hold:

- *(coverage)* for each $s \in S$, there exists $s' \in S'$ such that $(s, s') \in U$, and
- *(correctness)* for each $(t, t') \in U$, it holds (for each $y \in \Sigma^*$) that

$$(t, y) \in Q \Leftrightarrow (t', r(t, t', y)) \in Q'.$$

We call the pair (U, r) a *slice reduction* from $Q[S]$ to $Q'[S']$.

In this definition, we understand the parameterization (π_1, π_2) to be the mapping that, for all $(s, s') \in U$ and $y \in \Sigma^*$, returns the pair (s, s') given the triple (s, s', y).

Theorem 4.4. (Transitivity of slice reducibility) Suppose that $Q_1[S_1]$ slice reduces to $Q_2[S_2]$ and that $Q_2[S_2]$ slice reduces to $Q_3[S_3]$. Then $Q_1[S_1]$ slice reduces to $Q_3[S_3]$.

The following theorem allows one to derive an FPT-reduction or an nu-FPT-reduction from a slice reduction.

Theorem 4.5. Suppose that a case problem $Q[S]$ slice reduces to another case problem $Q'[S']$. Then, it holds that $\mathsf{param}\text{-}Q[S]$ nu-FPT-reduces to $\mathsf{param}\text{-}Q'[S']$; if in addition S and S' are computable, then $\mathsf{param}\text{-}Q[S]$ FPT-reduces to $\mathsf{param}\text{-}Q'[S']$.

For each parameterized class C, we define case-C to be the set of case problems that contains a case problem $Q[S]$ if and only if there exists a case problem $Q'[S']$ such that

- $\mathsf{param}\text{-}Q'[S']$ is in C,
- $Q[S]$ slice reduces to $Q'[S']$, and
- S' is computable.

Proposition 4.6. Suppose that C is a parameterized class, and that $Q[S]$ is a case problem in case-C. Then, the problem $\mathsf{param}\text{-}Q[S]$ is in nu-C; if it is assumed additionally that S is computable, then the problem $\mathsf{param}\text{-}Q[S]$ is in C.

Let $Q[S]$ be a case problem and let C be a parameterized class. We say that $Q[S]$ is case-C-hard if there exists a case problem $Q^-[S^-]$ such that

- $\mathsf{param}\text{-}Q^-[S^-]$ is C-hard,
- $Q^-[S^-]$ slice reduces to $Q[S]$, and
- S^- is computable.

Proposition 4.7. Suppose that C is a parameterized class, and that $Q[S]$ is a case problem that is case-C-hard. Then, the problem $\mathsf{param}\text{-}Q[S]$ is non-uniformly C-hard; if it is assumed additionally that S is computable, then the problem $\mathsf{param}\text{-}Q[S]$ is C-hard.

5. Thickness

In this section, we define a measure of first-order formulas that we call *thickness*, which we will show is the crucial measure that determines whether or not a graph-like set of sentences is tractable. From a high-level viewpoint, the measure is defined in the following way. We first define a notion of *organized formula* and show that each formula is logically equivalent to a positive combination of organized formulas; we then define a notion of *layered formula* and show that each organized formula is logically equivalent to a layered formula. We then, for each layered formula ϕ, define its thickness (denoted by $\mathsf{thick}_l(\phi)$), and then naturally extend this definition to positive combinations of layered formulas, and hence to all formulas. A key property of thickness, which we prove (Theorem 5.9), is that there exists an algorithm that, given a formula ϕ, outputs an equivalent formula that uses at most $\mathsf{thick}(\phi)$ many variables.

Definition 5.1. We define the set of organized formulas inductively, as follows.

- Each atom and each negated atom is an organized formula.
- If each of $\phi_1, \ldots, \phi_n$ is an organized formula and $v \in \mathsf{free}(\phi_1) \cap \cdots \cap \mathsf{free}(\phi_n)$, then $\exists v(\bigwedge_{i=1}^n \phi_i)$ and $\forall v(\bigvee_{i=1}^n \phi_i)$ are organized formulas.

Theorem 5.2. There exists an algorithm org^+ that, given a formula ϕ as input, outputs a positive combination $\mathrm{org}^+(\phi)$ of organized formulas that is logically equivalent to ϕ and that is in the syntactic closure of ϕ.

The algorithm of this theorem is defined recursively with respect to formula structure.

In what follows, when V is a set of variables and $Q \in \{\exists, \forall\}$ is a quantifier, we will use QV as shorthand for $Qv_1 \ldots Qv_n$, where $v_1, \ldots, v_n$ is a list of the elements of V. Our discussion will always be independent of the particular ordering chosen. Relative to a hypergraph H and a subset $S \subseteq V(H)$, we consider a set of edges $\{e_1, \ldots, e_k\}$ to be *S-connected* if one has connectedness of the graph with vertices $\{e_1, \ldots, e_k\}$ and having an edge between e_i and e_j if and only if $S \cap e_i \cap e_j \neq \emptyset$; we say that the hypergraph H is itself *S-connected* if $E(H)$ is S-connected.

Definition 5.3. We define the sets of *$\exists$-layered formulas* and of *$\forall$-layered formulas* to be the variable-loose formulas that can be constructed inductively, as follows:

- Each atom and each negated atom is both an $\exists$-layered formula and a $\forall$-layered formula.
- If each of $\phi_1, \ldots, \phi_n$ is a $\forall$-layered formula, and $X \subseteq \mathsf{free}(\phi_1) \cup \cdots \cup \mathsf{free}(\phi_n)$ is such that the hypergraph $\{\mathsf{free}(\phi_1), \ldots, \mathsf{free}(\phi_n)\}$ is X-connected, then $\exists X(\bigwedge_{i=1}^n \phi_i)$ is an $\exists$-layered formula.
- If each of $\phi_1, \ldots, \phi_n$ is an $\exists$-layered formula, and $Y \subseteq \mathsf{free}(\phi_1) \cup \cdots \cup \mathsf{free}(\phi_n)$ is such that the hypergraph $\{\mathsf{free}(\phi_1), \ldots, \mathsf{free}(\phi_n)\}$ is Y-connected, then $\forall Y(\bigvee_{i=1}^n \phi_i)$ is a $\forall$-layered formula.

Theorem 5.4. There exists an algorithm that, given an organized formula ϕ as input, outputs a layered formula that is logically equivalent to ϕ.

The intuitive idea behind the algorithm of Theorem 5.4 is to combine together (into a set quantification QV) quantifiers of the same type that occur adjacently in ϕ.

Theorem 5.5. There exists an algorithm lay^+ that, given a formula ϕ as input, outputs a positive combination $\mathrm{lay}^+(\phi)$ of layered formulas that is logically equivalent to ϕ.

Proof. Immediate from Theorems 5.2 and 5.4. □

Definition 5.6. We define the following measures on layered formulas.

- When ϕ is an atom or a negated atom, we define $\mathsf{thick}_l(\phi) = |\mathsf{free}(\phi)|$.
- Suppose that ϕ is a layered formula of the form $\exists U(\bigwedge_{i=1}^n \phi_i)$ or $\forall U(\bigvee_{i=1}^n \phi_i)$.
  We define the *local thickness* of ϕ as
  $$\mathsf{local\text{-}thick}_l(\phi) = 1 + \mathsf{tw}(\{\mathsf{free}(\phi_i) \mid i \in \underline{n}\} \cup \{\mathsf{free}(\phi)\})$$
  where the object to which the treewidth is applied is a hypergraph specified by its edge set, which hypergraph has vertex set $\cup_{i=1}^n \mathsf{free}(\phi_i)$.
  We define the *thickness* of ϕ inductively as
  $$\mathsf{thick}_l(\phi) = \max(\{\mathsf{local\text{-}thick}_l(\phi)\} \cup \{\mathsf{thick}_l(\phi_i) \mid i \in \underline{n}\}).$$
  We define the *quantified thickness* of ϕ as
  $$\mathsf{quant\text{-}thick}_l(\phi) = 1 + \mathsf{tw}(\{\mathsf{free}(\phi_i) \cap U \mid i \in \underline{n}\}).$$

Definition 5.7. The *thickness* of an arbitrary formula ϕ is defined as follows: let Ψ be the set of layered formulas that are positively combined subformulas of $\mathrm{lay}^+(\phi)$ such that $\mathrm{lay}^+(\phi)$ is a positive combination over Ψ; then,
$$\mathsf{thick}(\phi) = \max_{\psi \in \Psi} \mathsf{thick}_l(\psi).$$

Proposition 5.8. The function $\mathsf{thick}(\cdot)$ is computable.

Proof. This follows from the computability of $\mathrm{lay}^+(\phi)$ from ϕ (Theorem 5.5) and the definition of thickness. □

We indeed now demonstrate a principal property of thickness, namely, that this measure provides an upper bound on the number of variables needed to express a formula; this upper bound is effective in that there is an algorithm that computes an equivalent formula using a bounded number of variables. When $k \geqslant 1$, let us say that a formula *uses k many variables* if the set containing all variables that occur in the formula has size less than or equal to k.

Theorem 5.9. There exists an algorithm that, given as input a formula ϕ, outputs an equivalent formula that uses $\mathsf{thick}(\phi)$ many variables.

Definition 5.10. Let ϕ be a formula of the form $\exists V(\bigwedge_{i=1}^n \phi_i)$ or $\forall V(\bigvee_{i=1}^n \phi_i)$. Let us say that a pair $e = ((v_1, \ldots, v_m), E)$ consisting of an ordering $v_1, \ldots, v_m$ of the elements in $\bigcup_{i=1}^n \mathsf{free}(\phi_i)$ and a subset $E \subseteq K(\{v_1, \ldots, v_m\})$ is an *elimination ordering of* ϕ if:

- the variables in $\mathsf{free}(\phi)$ occur first in the ordering, that is, $\{v_1, \ldots, v_{|\mathsf{free}(\phi)|}\} = \mathsf{free}(\phi)$; and,
- e is an elimination ordering of the hypergraph $\{\mathsf{free}(\phi)\} \cup \{\mathsf{free}(\phi_i) \mid i \in \underline{n}\}$.

The following lemma can be taken as a variation of a lemma of Kolaitis and Vardi [15, Lemma 5.2].

Lemma 5.11. There exists a polynomial-time algorithm that, given a formula ϕ of the form $\exists V(\bigwedge_{i=1}^n \phi_i)$ or $\forall V(\bigvee_{i=1}^n \phi_i)$ and an elimination ordering e of ϕ, outputs a formula ϕ' that is logically equivalent to ϕ such that $\mathsf{width}(\phi') \leqslant \max(\{1 + \mathsf{lower\text{-}deg}(e)\} \cup \{\mathsf{width}(\phi_i) \mid i \in \underline{n}\})$.

Proof. (Theorem 5.9) By definition of $\mathsf{thick}(\phi)$ and by the computability of $\mathrm{lay}^+(\phi)$ from ϕ (Theorem 5.5), it suffices to prove that there is an algorithm that, given a layered formula ϕ, returns an equivalent formula that uses $\mathsf{thick}_l(\phi)$ many variables.

Consider the following algorithm A defined recursively on layered formulas. When ϕ is an atom or a negated atom, $A(\phi) = \phi$. When ϕ is of the form $\exists V(\bigwedge_{i=1}^n \phi_i)$ or $\forall V(\bigvee_{i=1}^n \phi_i)$, the algorithm proceeds as follows. Set H to be the hypergraph
$$\{\mathsf{free}(\phi_i) \mid i \in \underline{n}\} \cup \{\mathsf{free}(\phi)\}.$$
We have $1 + \mathsf{tw}(H) = \mathsf{local\text{-}thick}_l(\phi) \leqslant \mathsf{thick}_l(\phi)$. By calling the algorithm of Proposition 2.3 on the hypergraph H and the distinguished edge $\mathsf{free}(\phi)$, we can obtain an elimination ordering e of ϕ having $1 + \mathsf{lower\text{-}deg}(e) = 1 + \mathsf{tw}(H) \leqslant \mathsf{thick}_l(\phi)$. Let ϕ' be the formula obtained from ϕ by replacing each formula ϕ_i by $A(\phi_i)$; for each $i \in \underline{n}$, we have $\mathsf{width}(\phi_i) \leqslant \mathsf{thick}_l(\phi_i)$. By applying the algorithm of Lemma 5.11 to ϕ' and e, we obtain a formula ϕ'' having $\mathsf{width}(\phi'') \leqslant \mathsf{thick}_l(\phi)$. By renaming variables in ϕ'', we can obtain an equivalent formula where the number of variables used is equal to $\mathsf{width}(\phi'')$. The output $A(\phi)$ of the algorithm is this equivalent formula. □

6. Graph-like queries

In this section, we state our main theorem and two corollaries thereof, which describe the tractable graph-like sentence sets. In the following two sections, we prove the hardness portion of the main theorem.

Definition 6.1. Define MC to be the language of pairs
$$\{(\phi, \mathbf{B}) \mid \mathbf{B} \models \phi\}$$
where ϕ denotes a first-order sentence, and $\mathbf{B}$ denotes a relational structure.

Theorem 6.2. (Main theorem) Let Φ be a graph-like set of sentences having bounded arity. If Φ has bounded thickness, then the case problem $\mathsf{MC}[\Phi]$ is in case-FPT; otherwise, the case problem $\mathsf{MC}[\Phi]$ is case-W[1]-hard or case-co-W[1]-hard.

We give a proof of this theorem that makes a single forward reference to the main theorem of Section 8.

Proof. Suppose Φ has thickness bounded above by k. Define S' to be the set of sentences having thickness less than or equal to k. The set S' is computable by Proposition 5.8. We have that $\mathsf{param\text{-}MC}[S']$ is in FPT via the algorithm that, given an instance $(\phi, \mathbf{B})$, first checks if $\phi \in S'$, and if so, invokes Theorem 5.9 to obtain ϕ', and then performs the natural bottom-up, polynomial-time evaluation of ϕ' on $\mathbf{B}$ (à la Vardi [17]). The case problem $\mathsf{MC}[\Phi]$ slice reduces to $\mathsf{MC}[S']$ via the slice reduction $(\{(s', s') \mid s' \in S'\}, \pi_3)$.

Suppose that Φ has unbounded thickness. Theorem 8.1 yields that either case-CLIQUE or case-co-CLIQUE slice reduces to $\mathsf{MC}[\Phi]$. It then follows by definition that $\mathsf{MC}[\Phi]$ is case-W[1]-hard or case-co-W[1]-hard, since the problems CLIQUE and co-CLIQUE are W[1]-hard and co-W[1]-hard, respectively. □

We now provide two corollaries that describe the complexity of the problems $\mathsf{param\text{-}MC}[\Phi]$ addressed by the main theorem; the first corollary assumes that Φ is computable, while the second corollary makes no computability assumption on Φ. Both of these corollaries follow directly from Theorem 6.2 via use of Propositions 4.6 and 4.7.

Corollary 6.3. Let Φ be a computable, graph-like set of sentences having bounded arity. If Φ has bounded thickness, then the problem

$\mathsf{param\text{-}MC}[\Phi]$ is in FPT; otherwise, the problem $\mathsf{param\text{-}MC}[\Phi]$ is not in FPT, unless $\mathsf{W}[1] \subseteq \mathsf{FPT}$.

Corollary 6.4. Let Φ be a graph-like set of sentences having bounded arity. If Φ has bounded thickness, then the problem $\mathsf{param\text{-}MC}[\Phi]$ is in nu-FPT; otherwise, the problem $\mathsf{param\text{-}MC}[\Phi]$ is not in nu-FPT, unless $\mathsf{W}[1] \subseteq \mathsf{nu\text{-}FPT}$.

In the full version of this article, we provide a discussion of how the main theorem of the present paper can be used to readily derive the dichotomy theorem of graphical sets of quantified conjunctive queries [5]; a dual argument yields the corresponding theorem on graphical sets of quantified disjunctive queries. Note that it follows immediately from this discussion and [5, Example 3.5] that there exists a set of graph-like sentences having bounded arity that is tractable, but not contained in one of the tractable classes identified by Adler and Weyer [1].

Let us also note here that, as pointed out by Adler and Weyer, it is undecidable, given a first-order sentence ϕ and a value $k \geqslant 1$, whether or not ϕ is logically equivalent to a k-variable sentence, and hence one cannot expect an algorithm that takes a first-order sentence and outputs an equivalent one that minimizes the number of variables. (This undecidability result holds even for positive first-order logic [3].)

7. Accordion reductions

In this section, we introduce a notion that we call *accordion reduction*. When $C \subseteq \Sigma^* \times \Sigma^*$ is a set of string pairs and $S \subseteq \Sigma^*$ is a set of strings, we use $\mathsf{closure}_C(S)$ to denote the intersection of all sets T containing S and having the closure property that, if $(u, u') \in C$ and $u' \in T$, then $u \in T$. When an accordion reduction exists for such a C (with respect to a language Q), the theorem of this section (Theorem 7.2) yields that the problem $Q[\mathsf{closure}_C(S)]$ slice reduces to $Q[S]$. Hence, an accordion reduction is not itself a slice reduction, but its existence provides a sufficient condition for the existence of a class of slice reductions.

How is this section's theorem proved? One component of an accordion reduction is an FPT-computable mapping r that, for each $(u, u') \in C$, maps a Q-instance (u, y) to a Q-instance (u', y'). Intuitively, to give a slice reduction from $Q[\mathsf{closure}_C(S)]$ to $Q[S]$, one needs to reduce, for any $s \in S$ and $s_1 \in \mathsf{closure}_C(S)$, instances of the form $(s_1, \cdot)$ to instances of the form $(s, \cdot)$. The containment $s_1 \in \mathsf{closure}_C(S)$ implies the existence of a sequence $s_1, \ldots, s_k = s$ such that every pair (s_i, s_{i+1}) is in C. This naturally suggests applying the map r repeatedly, but note that there is no constant bound on the length k of the sequence. Hence, r needs to be sufficiently well-behaved so that, when composed with itself arbitrarily many times (in the described way), the end effect is that of an FPT-computable function that may serve as the map in the definition of slice reduction. (The author is mentally reminded of the closing of an accordion in thinking that this potentially long sequence of compositions yields a single well-behaved map.) To ensure this well-behavedness, we impose a condition that we call *measure-linearity*.

In the context of accordion reductions, a *measure* is a mapping $m : \Sigma^* \times \Sigma^* \to \mathbb{N}$ such that there exist a function $f : \Sigma^* \to \mathbb{N}$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ whereby, for all pairs $(s, y) \in \Sigma^* \times \Sigma^*$, it holds that $m(s, y) \leqslant |y| \leqslant f(s)p(m(s, y))$.

Definition 7.1. Let $Q \subseteq \Sigma^* \times \Sigma^*$ be a language of pairs. With respect to Q, an *accordion reduction* consists of:

- a computably enumerable language $C \subseteq \Sigma^* \times \Sigma^*$,
- a measure $m : \Sigma^* \times \Sigma^* \to \mathbb{N}$, and
- a mapping $r : \Sigma^* \times \Sigma^* \times \Sigma^* \to \Sigma^*$ that has $\mathsf{dom}(r) = C \times \Sigma^*$, that is FPT-computable with respect to the parameterization (π_1, π_2), and that is *measure-linear* in that, for each pair $(u, u') \in C$, there exists a constant $B_{(u,u')}$ such that, for each $y \in \Sigma^*$, it holds that $m(u', r(u, u', y)) \leqslant B_{(u,u')} m(u, y)$,

such that the following condition holds:

- *(correctness)* for each $(u, u') \in C$, it holds (for each $y \in \Sigma^*$) that

$$(u, y) \in Q \Leftrightarrow (u', r(u, u', y)) \in Q.$$

Theorem 7.2. Suppose that $Q[S]$ is a case problem, and that (C, m, r) is an accordion reduction with respect to Q. Then, the case problem $Q[\mathsf{closure}_C(S)]$ slice reduces to the case problem $Q[S]$.

8. Hardness

In this section, we establish the main intractability result of the paper, namely, that the case problem $\mathsf{MC}[\Phi]$ is hard when Φ is graph-like and has unbounded thickness.

Theorem 8.1. Suppose that Φ is a set of graph-like sentences of bounded arity such that $\mathsf{thick}(\Phi)$ is unbounded. Then, either case-CLIQUE or case-co-CLIQUE slice reduces to $\mathsf{MC}[\Phi]$.

Proof. Immediate from Lemmas 8.2 and 8.3, and Theorem 8.5. □

This intractability result is obtained by composing three slice reductions. Define a formula to be *friendly* if it is loose, positive, and layered. We first show (Lemma 8.2) that there exists a set of friendly sentences Ψ, with $\mathsf{thick}_l(\Psi)$ unbounded, such that $\mathsf{MC}[\Psi]$ slice reduces to $\mathsf{MC}[\Phi]$. We next show (Lemma 8.3) that a multi-sorted version $\mathsf{full}(\Psi)$ of Ψ has the property that $\mathsf{MC}_s[\mathsf{full}(\Psi)]$ slice reduces to $\mathsf{MC}[\Psi]$; here, MC_s denotes the multi-sorted generalization of MC. Finally, we directly slice reduce either case-CLIQUE or case-co-CLIQUE to $\mathsf{MC}_s[\Psi]$ (Theorem 8.5); this third reduction is obtained via an accordion reduction.

Lemma 8.2. Suppose that Φ is a set of graph-like sentences of bounded arity such that $\mathsf{thick}(\Phi)$ is unbounded. There exists a set of friendly sentences Ψ such that $\mathsf{MC}[\Psi]$ slice reduces to $\mathsf{MC}[\Phi]$ and such that $\mathsf{thick}_l(\Psi)$ is unbounded.

In what follows, we will work with multi-sorted relational first-order logic, formalized as follows. (For differentiation, we will refer to formulas in the usual first-order logic considered thus far as *one-sorted*.) A *signature* is a pair $(\sigma, \mathcal{S})$ where $\mathcal{S}$ is a set of *sorts* and σ is a set of relation symbols; each relation symbol $R \in \sigma$ has an associated arity $\mathrm{ar}(R)$ which is an element of $\mathcal{S}^*$. In a formula over signature $(\sigma, \mathcal{S})$, each variable v has associated with it a sort $s(v)$ from $\mathcal{S}$; an atom is a formula $R(v_1, \ldots, v_k)$ where $R \in \sigma$ and $s(v_1) \ldots s(v_k) = \mathrm{ar}(R)$. A structure $\mathbf{B}$ on signature $(\sigma, \mathcal{S})$ consists of an $\mathcal{S}$-sorted family $\{B_s \mid s \in \mathcal{S}\}$ of sets called the *universe* of $\mathbf{B}$, and for each symbol $R \in \sigma$, an interpretation $R^{\mathbf{B}} \subseteq B_{\mathrm{ar}(R)}$, where for a word $w = w_1 \ldots w_k \in \mathcal{S}^*$, we use B_w to denote the product $B_{w_1} \times \cdots \times B_{w_k}$. We use MC_s to denote the multi-sorted version of MC, that is, it is the language of pairs $(\phi, \mathbf{B})$ where ϕ is a sentence and $\mathbf{B}$ is a structure both having the same signature $(\sigma, \mathcal{S})$, and $\mathbf{B} \models \phi$.

Suppose that ϕ is a multi-sorted friendly formula on signature $(\sigma, \mathcal{S})$, and let V be the set of variables occurring in ϕ; we say that ϕ is *fully-sorted* if $V \subseteq \mathcal{S}$ and for each $v \in V$, the sort of v is v itself (that is, $s(v) = v$). When ψ is a one-sorted friendly formula, and V is the set of variables that occur in ψ, we use $\mathsf{full}(\psi)$ to denote the natural fully-sorted formula induced by ψ, namely, the formula on signature (σ, V) where σ contains those symbols occurring in ψ, and, if $R(v_1, \ldots, v_k)$ appears in ψ, then $\mathrm{ar}(R) = v_1 \ldots v_k$ (this is well-defined since ϕ is symbol-loose).

Lemma 8.3. Let Ψ be a set of one-sorted friendly sentences. The set $\mathsf{full}(\Psi)$ of fully-sorted friendly sentences has the property that $\mathsf{MC}_s[\mathsf{full}(\Psi)]$ slice reduces to $\mathsf{MC}[\Psi]$.

Proof. We give a slice reduction. Define U to be the set that contains each pair of the form $(\mathsf{full}(\psi), \psi)$. Suppose that ψ and $\mathbf{B}$ are over signature $(\sigma, \mathcal{S})$. Define $r(\mathsf{full}(\psi), \psi, \mathbf{B}) = \mathbf{B}'$, where $\mathbf{B}'$ is defined as follows. Let B' be a set whose cardinality is $\max_{s \in \mathcal{S}} |B_s|$. For each sort $s \in \mathcal{S}$, fix a map $f_s : B' \to B_s$ that is surjective. For each relation symbol $R \in \sigma$ of arity $s_1 \ldots s_k$, define $R^{\mathbf{B}'} = \{(b'_1, \ldots, b'_k) \in B'^k \mid (f_{s_1}(b'_1), \ldots, f_{s_k}(b'_k)) \in R^{\mathbf{B}}\}$. It is straightforward to prove by induction that, for all subformulas ϕ of ψ, and each assignment g from the set of variables to B', that $\mathbf{B}', g \models \phi$ if and only if $\mathbf{B}, f \circ^s g \models \mathsf{full}(\phi)$. Here, $f \circ^s g$ denotes the mapping that sends each variable v to $f_{s(v)}(g(v))$ and we view $\mathsf{full}(\phi)$ as a formula over the signature $(\sigma, \mathcal{S})$ of ψ. The correctness of the reduction follows. □

In the remainder of this section, we assume that all formulas and structures under discussion are multi-sorted.

Let us say that a friendly formula is a *simple formula* if it is of the form $\exists X(\bigwedge_{i=1}^n \alpha_i)$ or of the form $\forall Y(\bigvee_{i=1}^n \alpha_i)$ where each α_i is an atom. When ϕ is a simple formula where the variables V are those that are quantified initially, we say that ψ is *a sentence based on* ϕ if ψ is a simple sentence derivable from ϕ by replacing each atom $R(w_1, \ldots, w_k)$ with an atom whose variables are the elements in $\{w_1, \ldots, w_k\} \cap V$.

We now present an accordion reduction that will be used to derive our hardness result. When Ψ is a set of friendly sentences, this accordion reduction will allow a simple subformula $\phi' = \exists V \chi$ to simulate a disjunction of atoms on $\mathsf{free}(\phi')$, and likewise for a simple subformula $\phi' = \forall V \chi$ to simulate a corresponding conjunction; this is made precise as follows. The set C is defined to contain a pair (ψ, ϕ) of friendly sentences if there exists a simple subformula $\phi' = QV\chi$ of ϕ such that one of the following conditions holds:

(1) ψ is a sentence based on ϕ'.

(2) $Q = \exists$ and ψ is a friendly sentence obtained from ϕ by replacing ϕ' with $\bigvee_{i=1}^m E_i(v_{i1}, v_{i2})$, where the tuples (v_{i1}, v_{i2}) are such that $\{v_{11}, v_{12}\}, \ldots, \{v_{m1}, v_{m2}\}$ is a list of the elements in $K(\mathsf{free}(\phi'))$ and the E_i are relation symbols (each of which is fresh in that it does not appear elsewhere in ψ).

(3) $Q = \forall$ and ψ is a friendly sentence obtained from ϕ by replacing ϕ' with $\bigwedge_{i=1}^m E_i(v_{i1}, v_{i2})$ where the tuples (v_{i1}, v_{i2}) and the symbols E_i are as described in the previous case.

Theorem 8.4. Let M be the measure such that $M(\phi, \mathbf{B})$ is equal to $\max_{s \in \mathcal{S}} |B_s|$ when $\mathbf{B}$ is a multi-sorted structure defined on the signature $(\sigma, \mathcal{S})$ of ϕ, and such that $M(\phi, \mathbf{B})$ is equal to 0 otherwise. There exists a mapping $r : \Sigma^* \times \Sigma^* \times \Sigma^* \to \Sigma^*$ such that the triple (C, M, r) is an accordion reduction with respect to MC_s. (Here, C is the set defined above.)

Proof. Suppose $(\psi, \phi, \mathbf{A})$ is a triple with $(\psi, \phi) \in C$ and where $\mathbf{A}$ is a structure over the signature of ψ. We define $r(\psi, \phi, \mathbf{A})$ to be the structure $\mathbf{B}$, defined as follows.

It is straightforward to treat the case where there exists a simple subformula ϕ' of ϕ such that ψ is the sentence based on ϕ' (here, we omit discussion of this case).

In the remainder of this proof, we consider the case that there exist a simple subformula $\phi' = \exists V \chi$ of ϕ such that ψ is a friendly sentence obtained from ϕ by replacing ϕ' with a disjunction ψ' as in the definition of C. (Dual to this case is the remaining case where $\phi' = \forall V \chi$ and ψ is obtained from ϕ by replacing ϕ' with a conjunction.) We will denote the disjunction ψ' by $E_1(w_{11}, w_{12}) \vee \cdots \vee E_m(w_{m1}, w_{m2})$. We define the universe of $\mathbf{B}$ as follows.

- For each sort u of $\mathbf{A}$, we define $B_u = A_u$.
- For each $v \in V$, define $B_v = \{(E_\ell, u, a) \mid \ell \in \underline{m}, u \in \{w_{\ell 1}, w_{\ell 2}\}, a \in A_u\}$

As $m \leqslant |V|^2$ and a variable u appearing as the second coordinate in an element of a set B_v must be an element of $\mathsf{free}(\phi')$, We have $M(\phi, \mathbf{B}) \leqslant |V|^2 \cdot |\mathsf{free}(\phi')| \cdot M(\psi, \mathbf{A})$; this confirms measure-linearity of M, since $|V|$ and $|\mathsf{free}(\phi')|$ are functions of the pair (ψ, ϕ).

For each atom $R(u_1, \ldots, u_k)$ in ϕ that occurs outside of ϕ' (equivalently, that also appears in ψ), define $R^{\mathbf{B}} = R^{\mathbf{A}}$.

For each atom $R(u_1, \ldots, u_k)$ that occurs in ϕ', define $R^{\mathbf{B}}$ to contain a tuple $(b_1, \ldots, b_k) \in B_{u_1 \ldots u_k}$ if and only if the following two conditions hold:

- For all $i, j \in \underline{k}$, if $u_i \in V$ and $u_j \in V$, then $b_i = b_j$.
- For all $i, j \in \underline{k}$, if $u_i \in V$, $b_i = (E_\ell, u, a)$, and $u_j \notin V$ (equivalently, $u_j \in \mathsf{free}(\phi')$), then
  - $u_j = u$ implies $b_j = a$, and
  - $\{u_j, u\} = \{w_{\ell 1}, w_{\ell 2}\}$ implies $\mathbf{A}, \{(u, a), (u_j, b_j)\} \models E_\ell(w_{\ell 1}, w_{\ell 2})$. (Here, we use a set of pairs to denote a partial map.)

To verify that $\mathbf{A} \models \psi$ if and only if $\mathbf{B} \models \phi$, it suffices to verify that, for any assignment f defined on $\mathsf{free}(\phi') = \mathsf{free}(\psi')$ taking each variable u to an element of A_u, that

$$\mathbf{A}, f \models \psi' \Leftrightarrow \mathbf{B}, f \models \phi'.$$

($\Rightarrow$): There exists $\ell \in \underline{m}$ such that $\mathbf{A}, f \models E_\ell(w_{\ell 1}, w_{\ell 2})$. Pick w to be a variable in $\{w_{\ell 1}, w_{\ell 2}\}$, and consider the extension f^+ of f that sends each variable in V to $(E_\ell, w, f(w))$. It is straightforward to verify that $\mathbf{B}, f^+$ satisfies the conjunction of ϕ'; we do so as follows. Suppose that $R(u_1, \ldots, u_k)$ is an atom in this conjunction. We claim that $(f^+(u_1), \ldots, f^+(u_k)) \in R^{\mathbf{B}}$. This tuple clearly satisfies the first condition in the definition of $R^{\mathbf{B}}$; to check the second condition, suppose that $i, j \in \underline{k}$ are such that $u_i \in V$ and $u_j \notin V$. We have $f^+(u_i) = (E_\ell, w, f(w))$. If $u_j = w$, then indeed $f^+(u_j) = f(w)$. If $\{u_j, w\} = \{w_{\ell 1}, w_{\ell 2}\}$, then $\{(w, f(w)), (u_j, f^+(u_j))\}$ is equal to $f \upharpoonright \{w_{\ell 1}, w_{\ell 2}\}$, and we have $\mathbf{A}, f \upharpoonright \{w_{\ell 1}, w_{\ell 2}\} \models E_\ell(w_{\ell 1}, w_{\ell 2})$ by our choice of ℓ.

($\Leftarrow$): Suppose that $\mathbf{B}, f^+$ satisfies the conjunction of ϕ'. By the definition of $\mathbf{B}$ and since ϕ' is a layered formula, f^+ maps all variables in V to the same value (E_ℓ, u, a). There exists an atom $R(u_1, \ldots, u_k)$ in ϕ' such that one of its variables u_j has the property that $\{u_j, u\} = \{w_{\ell 1}, w_{\ell 2}\}$. It follows, from the definition of $R^{\mathbf{B}}$, that $\mathbf{A}, f^+ \upharpoonright \{u, u_j\} \models E_\ell(w_{\ell 1}, w_{\ell 2})$. □

Theorem 8.5. Let Ψ be a set of fully-sorted, friendly sentences such that $\mathsf{thick}_l(\Psi)$ is unbounded. Then, either case-CLIQUE or case-co-CLIQUE slice reduces to $\mathsf{MC}_s[\Psi]$.

Suppose that ψ is a simple friendly formula that occurs as a subformula of a formula ϕ. We say that ψ is an *existential k-clique* if it is of the form $\exists X(\bigwedge_{i=1}^n \alpha_i)$ and there exists a set V of variables, with $|V| \geqslant k$, that are existentially quantified (in ϕ) such that for each set $\{v, v'\} \in K(V)$, there exists an atom α_i with $\{v, v'\} \subseteq \mathsf{free}(\alpha_i)$. We define a *universal k-clique* dually.

Proof. Let C be as defined above in the discussion. By appeal to Theorem 8.4, it suffices to prove that either case-CLIQUE or case-co-CLIQUE slice reduces to $\mathsf{MC}_s[\mathsf{closure}_C(\Psi)]$. When Θ is a set of layered sentences, let us use the term *Θ-relevant subformula* to refer to a layered subformula ϕ of a sentence θ in Θ. For each Ψ-relevant subformula ϕ, it holds that $\mathsf{local\text{-}thick}_l(\phi) \leqslant$

$\mathsf{quant\text{-}thick}_l(\phi) + |\mathsf{free}(\phi)|$ (this is since the hypergraph from which $\mathsf{quant\text{-}thick}_l(\phi)$ is defined is the hypergraph from which $\mathsf{local\text{-}thick}_l(\phi)$ is defined, but with the vertices in $\mathsf{free}(\phi)$ removed). By the definition of $\mathsf{thick}_l(\cdot)$, the quantity $\mathsf{local\text{-}thick}_l(\phi)$ over Ψ-relevant subformulas ϕ is unbounded. Thus, over Ψ-relevant subformulas ϕ, either the quantity $\mathsf{quant\text{-}thick}_l(\phi)$ or the quantity $|\mathsf{free}(\phi)|$ is unbounded. We may therefore consider two cases.

Assume that $|\mathsf{free}(\phi)|$ is unbounded over Ψ-relevant subformulas ϕ. By conditions (2) and (3) in the definition of C, we obtain that $|\mathsf{free}(\phi)|$ is unbounded over simple $\mathsf{closure}_C(\Psi)$-relevant subformulas ϕ. We assume that $|\mathsf{free}(\phi)|$ is unbounded over such simple subformulas ϕ that use existential quantification (if not, then $|\mathsf{free}(\phi)|$ is unbounded over such simple subformulas ϕ that use universal quantification, and the argumentation is dual). We now consider two cases. If the number of universally quantified variables in $\mathsf{free}(\phi)$ is unbounded over such subformulas ϕ, then by condition (2) of the definition of C applied to each such subformula (and the sentence in which it appears), we obtain that, for each $k \geqslant 1$, the set $\mathsf{closure}_C(\Psi)$ contains a sentence that contains a universal k-clique. Otherwise, the number of existentially quantified variables in $\mathsf{free}(\phi)$ is unbounded over such subformulas ϕ; in this case, by an application of condition (3) followed by an application of condition (2) (again, to each such subformula ϕ and the sentence in which it appears), we obtain that, for each $k \geqslant 1$, the set $\mathsf{closure}_C(\Psi)$ contains a sentence that contains an existential k-clique. In this latter case, let us explain how to exhibit a slice reduction from $\mathsf{case\text{-}CLIQUE}$ to $\mathsf{MC}_s[\mathsf{closure}_C(\Psi)]$ (in the former case, one dually obtains a reduction from $\mathsf{case\text{-}co\text{-}CLIQUE}$ to $\mathsf{MC}_s[\mathsf{closure}_C(\Psi)]$). We define U to be the set of pairs (k, θ) such that $k \geqslant 1$ and θ is a sentence that contains an existential k-clique as a subformula. Let us define $r(k, \theta, (V, E))$ to be the structure $\mathbf{B}$ described as follows. Let W be the set of variables that witnesses the existential k-clique. For each relation symbol R of an atom in θ that does not witness the existential k-clique, we may set $R^{\mathbf{B}}$ to either $\varnothing$ or B^r (here, r is the arity of R) in such a way that $\mathbf{B} \models \theta$ if and only if $\mathbf{B} \models \exists W(\wedge_j \beta_j)$, where the β_j are the atoms witnessing the existential k-clique. By defining, for each remaining relation symbol F (that is, for each relation symbol F of an atom β_j), the relation $F^{\mathbf{B}}$ to be E, we obtain that (V, E) contains a k-clique if and only if $\mathbf{B} \models \theta$.

Now assume that $\mathsf{quant\text{-}thick}_l(\phi)$ is unbounded over Ψ-relevant subformulas ϕ. By conditions (2) and (3) in the definition of C, we obtain that $\mathsf{quant\text{-}thick}_l(\phi)$ is unbounded over simple $\mathsf{closure}_C(\Psi)$-relevant subformulas ϕ. By considering the sentences based on these subformulas (and by invoking condition (1) in the definition of C), we obtain that $\mathsf{closure}_C(\Psi)$ contains simple sentences $\exists X(\bigwedge \alpha_i)$ or contains simple sentences $\exists Y(\bigvee \alpha_i)$ such that, over these sentences, $\mathsf{quant\text{-}thick}_l(\cdot)$ is unbounded. In the former case, the intractability result of Grohe, Schwentick and Segoufin [13] directly yields a slice reduction from $\mathsf{case\text{-}CLIQUE}$ to $\mathsf{MC}_s[\mathsf{closure}_C(\Psi)]$ where the set U contains a pair (k, θ) when $k \geqslant 1$ and θ is a sentence $\exists X(\bigwedge \alpha_i)$ whose corresponding graph admits a k-by-k grid as a minor. The latter case is dual (one obtains a slice reduction from $\mathsf{case\text{-}co\text{-}CLIQUE}$ to $\mathsf{MC}_s[\mathsf{closure}_C(\Psi)]$). □

Acknowledgments

The author was supported by the Spanish Project FORMALISM (TIN2007-66523), by the Basque Government Project S-PE12UN050(SAI12/219), and by the University of the Basque Country under grant UFI11/45. The author thanks Montserrat Hermo and Simone Bova for useful comments.

References

[1] I. Adler and M. Weyer. Tree-width for first order formulae. *Logical Methods in Computer Science*, 8(1), 2012.

[2] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[3] S. Bova and H. Chen. The complexity of width minimization for existential positive queries. In *ICDT*, pages 235–244, 2014.

[4] H. Chen. On the complexity of existential positive queries. *ACM Trans. Comput. Log.*, 15(1), 2014.

[5] H. Chen and V. Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. In *LICS*, 2012.

[6] H. Chen and M. Grohe. Constraint satisfaction with succinctly specified relations. *Journal of Computer and System Sciences*, 76(8):847–860, 2010.

[7] H. Chen and D. Marx. Block-sorted quantified conjunctive queries. In *ICALP*, 2013.

[8] H. Chen and M. Müller. The fine classification of conjunctive queries and parameterized logarithmic space complexity. In *PODS*, 2013.

[9] Y. Chen, M. Thurley, and M. Weyer. Understanding the complexity of induced subgraph isomorphisms. In *ICALP 2008*, pages 587–596, 2008.

[10] A. Durand and S. Mengel. Structural tractability of counting of solutions to conjunctive queries. In *Proceedings of the 16th International Conference on Database Theory (ICDT 2013)*, 2013.

[11] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[12] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.

[13] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *STOC 2001*, 2001.

[14] P. Jonsson, V. Lagerkvist, and G. Nordh. Blowing holes in various aspects of computational problems, with applications to constraint satisfaction. In *CP*, pages 398–414, 2013.

[15] P. Kolaitis and M. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences*, 61: 302–332, 2000.

[16] C. Papadimitriou and M. Yannakakis. On the Complexity of Database Queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

[17] M. Y. Vardi. On the complexity of bounded-variable queries. In *PODS'95*, pages 266–276, 1995.

Proof of Proposition 3.11

Proof. For (1), one can use the fact that $f_1(k)p_1(n) + \cdots + f_m(k)p_m(n)$ is bounded above by $(f_1(k) + \cdots + f_m(k))(p_1(n) + \cdots + p_m(n))$.

For (2), it suffices to observe that the product $(f_1(k)p_1(n)) \cdots (f_m(k)p_m(n))$ can be grouped as $(f_1(k) \cdots f_m(k))(p_1(n) \cdots p_m(n))$.

For (3), it suffices to observe that $q(f(k)p(n))$ is bounded above by $q(f(k))q(p(n))$; note that $q(p)$ is the composition of two polynomials, and hence itself a polynomial. □

Proof of Theorem 4.4

Proof. Let (U_1, r_1) be a slice reduction from $Q_1[S_1]$ to $Q_2[S_2]$, and let (U_2, r_2) be a slice reduction from $Q_2[S_2]$ to $Q_3[S_3]$. Define U to be the set

$$\{(t_1, t_3) \mid \text{ there exists } t_2 \in \Sigma^* \text{ such that } (t_1, t_2) \in U_1 \text{ and } (t_2, t_3) \in U_2\}.$$

It is straightforward to verify that U is computably enumerable and that there exists an algorithm A_U that, given a pair $(t_1, t_3) \in U$, outputs a value $t_2 \in \Sigma^*$ such that $(t_1, t_2) \in U_1$ and $(t_2, t_3) \in U_2$. Define r to be a partial function such that, when $(t_1, t_3) \in U$, for each $y \in \Sigma^*$ it holds that $r(t_1, t_3, y) = r_2(t_2, t_3, r_1((t_1, t_2), y))$; here, we use t_2 to denote $A_U(t_1, t_3)$.

We verify that (U, r) is a slice reduction from $Q_1[S_1]$ to $Q_3[S_3]$. For each $t_1 \in S_1$, there exists $t_2 \in S_2$ such that $(t_1, t_2) \in U_1$; and, for each $t_2 \in S_2$, there exists $t_3 \in S_3$ such that $(t_2, t_3) \in U_2$. Hence, for each $t_1 \in S_1$, there exists $t_3 \in S_3$ such that $(t_1, t_3) \in U$. This confirms the coverage condition; we now check correctness. Suppose that $(t_1, t_3) \in U$, and let $y \in \Sigma^*$. Set $y' = r_1(t_1, t_2, y)$. We have that

$$(t_1, y) \in Q_1 \Leftrightarrow (t_2, y') \in Q_2$$

and

$$(t_2, y') \in Q_2 \Leftrightarrow (t_3, r_2(t_2, t_3, y')) \in Q_3.$$

It follows that

$$(t_1, y) \in Q_1 \Leftrightarrow (t_3, r(t_1, t_3, y)) \in Q_3.$$

It remains to verify that the function r is FPT-computable with respect to the parameterization (π_1, π_2). We verify this by considering the natural algorithm at this point, namely, the following algorithm: given (t_1, t_3, y), invoke A_U on (t_1, t_3) and, if this computation halts, set t_2 to the result; then, compute $y' = r_1(t_1, t_2, y)$ using the algorithm witnessing FPT-computability, and finally, compute $r_2(t_2, t_3, y')$ using the algorithm witnessing FPT-computability. On an input $x = (t_1, t_3, y) \in U \times \Sigma^*$, the running time of this algorithm can be upper bounded by

$$F(t_1, t_3) + f_1(t_1, t_2)p_1(|(t_1, t_2, y)|) + f_2(t_2, t_3)p_2(|(t_2, t_3, y')|)$$

where $F(t_1, t_3)$ denotes the running time of A_U on input (t_1, t_3); and (f_1, p_1) and (f_2, p_2) witness the FPT-computability of r_1 and r_2, respectively; we here neglect additive constants. We need to show that this running time is degree-bounded with respect to (π_1, π_2); for the remainder of the proof, let us simply use *degree-bounded* to mean degree-bounded with respect to this parameterization. We view the various quantities under discussion as functions of $x = (t_1, t_3, y)$. Since $f_1(t_1, t_2)$ and $f_2(t_2, t_3)$ can be viewed as functions of (t_1, t_3), by Proposition 3.11 (1, 2), it suffices to verify that each of $p_1(|(t_1, t_2, y)|)$, $p_2(|(t_2, t_3, y')|)$ is degree-bounded. By appeal to Proposition 3.11(3), to verify that $p_1(|(t_1, t_2, y)|)$ is degree-bounded, it suffices to observe that each of $|t_1|$, $|t_2|$, $|y|$ is degree-bounded. Similarly, to verify that $p_2(|(t_2, t_3, y')|)$ is degree-bounded, it suffices to verify that each of $|t_2|$, $|t_3|$, and $|y'|$ are degree-bounded; this is clear for $|t_2|$ and $|t_3|$, and the size $|y'|$ is bounded above by $f_1(t_1, t_2)p_1(|(t_1, t_2, y)|)$, which is degree-bounded since we just verified that $p_1(|(t_1, t_2, y)|)$ is degree-bounded. □

Proof of Theorem 4.5

Proof. Let (U, r) be the slice reduction. First, consider the case where both S and S' are computable. Since S and S' are both computable and U is computably enumerable, there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that, for all $s \in S$, it holds that $(s, f(s)) \in U$. We define the reduction g so that $g(s, x)$ is equal to $(f(s), r(s, f(s), x))$ for all $(s, x) \in S \times \Sigma^*$, and is otherwise defined to be a fixed string outside of Q' (such a string exists by Assumption 3.1). The mapping g has a natural algorithm, namely, given (s, x), check if $s \in S$ (via an algorithm for S); if $s \notin S$, return a fixed string outside of Q', otherwise, compute the value of g using an algorithm for f and an algorithm witnessing FPT-computability of r. Suppose that $s \in S$; since r is FPT-computable with respect to (π_1, π_2), we have that the value $r(s, f(s), x)$, viewed as a function of (s, x), is FPT-computable with respect to π_1; the value $f(s)$ is, as well. Thus, the mapping g is FPT-computable with respect to π_1.

In the case that no computability assumptions are placed on S and S', there exists a function $f : \Sigma^* \to \Sigma^*$ such that, for all $s \in S$, it holds that $(s, f(s)) \in U$. The reduction g defined as above is readily verified to be nu-FPT-computable with respect to π_1, via an ensemble of algorithms where A_s contains as hard-coded information whether or not $s \in S$ and (if so) the value of $f(s)$. □

Proof of Proposition 4.6

Proof. There exists a case problem $Q'[S']$ satisfying the conditions given in the definition of case-C. Since $Q[S]$ slice reduces to $Q'[S']$, by Theorem 4.5, it holds that param-$Q[S]$ nu-FPT-reduces to param-$Q'[S']$. Since param-$Q'[S']$ is in C, it follows that param-$Q[S]$ is in nu-C. If in addition it is assumed that S is computable, by Theorem 4.5, we obtain that param-$Q[S]$ FPT-reduces to param-$Q'[S']$, and hence that param-$Q[S]$ is in C (by appeal to Assumption 3.6). □

Proof of Proposition 4.7

Proof. There exists a case problem $Q^-[S^-]$ satisfying the conditions given in the definition of case-C-hard. Since $Q^-[S^-]$ slice reduces to $Q[S]$, by Theorem 4.5, it holds that param-$Q^-[S^-]$ nu-FPT-reduces to param-$Q[S]$. By the C-hardness of param-$Q^-[S^-]$, each problem in C FPT-reduces to param-$Q^-[S^-]$, and hence, by Proposition 3.7, each problem in C nu-FPT-reduces to param-$Q[S]$. If in addition it is assumed that S is computable, by Theorem 4.5, the problem param-$Q^-[S^-]$ FPT-reduces to the problem param-$Q[S]$; from the C-hardness of param-$Q^-[S^-]$ and from Proposition 3.7, we obtain that param-$Q[S]$ is C-hard. □

Proof of Theorem 5.2

Proof. To give the algorithm, we first recursively define a procedure that, given a formula ϕ where negations appear only in front of atoms, outputs both a CNF of organized formulas and a DNF of organized formulas, each of which is equivalent to ϕ and in the syntactic closure of ϕ. We will make tacit use of transformation (α). Observe that it suffices to show that either such a CNF or such a DNF can be computed, since one can convert between such a CNF and such a DNF by use of transformation (δ). The procedure is defined as follows.

- When ϕ is an atom or a negated atom, the procedure returns ϕ.
- When ϕ has the form $\psi_1 \wedge \psi_2$, the procedure computes a CNF for ϕ by taking the conjunction of CNFs for ψ_1 and ψ_2, which

can be computed recursively. The case where ϕ has the form $\psi_1 \vee \psi_2$ is defined dually.

- When ϕ has the form $\exists x \psi$, the procedure performs the following. (The case where ϕ has the form $\forall y \psi$ is dual.) First, it recursively computes a DNF for ψ; denote the DNF by $\bigvee_i \psi_i$. By appeal to transformation (β), the formula ϕ is logically equivalent to $\bigvee_i (\exists x \psi_i)$. To obtain a DNF for ϕ, it thus suffices to show that each formula of the form $\exists x \psi_i$ can be transformed to an equivalent conjunction of organized formulas. Each formula ψ_i is a conjunction $\psi_i^1 \wedge \cdots \wedge \psi_i^k$ of organized formulas; write ψ_i as $\psi_i^x \wedge \psi_i^{-x}$, where ψ_i^x is the conjunction of the formulas ψ_i^j such that $x \in \mathsf{free}(\psi_i^j)$, and ψ_i^{-x} is the conjunction of the remaining formulas ψ_i^j. We have, by transformation (γ), that $\exists x \psi_i$ is equivalent to $(\exists x \psi_i^x) \wedge \psi_i^{-x}$.

The algorithm org^+, given a formula ϕ, computes an equivalent formula ϕ' where negations appear only in front of atoms (using transformations (ϵ)), and then invokes the described procedure on ϕ' and outputs either the CNF or the DNF returned by the procedure. □

Proof of Theorem 5.4

Proof. We define the algorithm recursively.

- If ϕ is an atom or a negated atom, the algorithm returns ϕ.
- If ϕ is an organized formula of the form $\exists v(\bigwedge_{i=1}^{\ell} \theta_i)$ with $v \in \mathsf{free}(\theta_1) \cap \cdots \cap \mathsf{free}(\theta_\ell)$, the algorithm proceeds as follows. By renaming variables if necessary, the algorithm ensures that no variable is quantified twice in ϕ, and also that no variable is both free and quantified in ϕ. For each i, the algorithm recursively computes, from θ_i, a logically equivalent layered formula θ_i'. The algorithm then writes ϕ as the formula
$$\exists v((\exists X_1 \phi_1) \wedge \cdots \wedge (\exists X_m \phi_m)) \wedge (\psi_1 \wedge \cdots \wedge \psi_n)$$
where $\exists X_1 \phi_1, \ldots, \exists X_m \phi_m$ is a list of the formulas θ_i' that begin with existential quantification, and where $\psi_1 \wedge \cdots \wedge \psi_n$ is a list of the remaining formulas θ_i'. Note that each ϕ_j is the conjunction of $\forall$-layered formulas, and each ψ_j is a $\forall$-layered formula.
For each $j \in \underline{m}$, let H_j denote the hypergraph of ϕ_j, that is, the hypergraph where an edge is present if it is the set of free variables of a conjunct of ϕ_j. We have that the hypergraph H_j is X_j-connected. Since each H_i contains v in an edge and it holds that $v \in \mathsf{free}(\psi_1) \cap \cdots \cap \mathsf{free}(\psi_n)$, we have that the hypergraph H with edge set
$$E(H_1) \cup \cdots \cup E(H_m) \cup \{\mathsf{free}(\psi_1)\} \cup \cdots \cup \{\mathsf{free}(\psi_n)\}$$
is $(X_1 \cup \cdots \cup X_m \cup \{v\})$-connected. Consider the conjunction of the ϕ_j and of the ψ_j. This conjunction can be viewed as the conjunction of $\forall$-layered formulas whose free variable sets are exactly the edges of H; denote this conjunction by χ. The $\exists$-layered formula $\exists(X_1 \cup \cdots \cup X_m \cup \{v\})\chi$ is logically equivalent to ϕ; this is because, by the variable renaming done initially, the variables in a set X_i do not appear in a formula ϕ_j when $j \neq i$, nor do they appear in a formula ψ_j.
- If ϕ is an organized formula of the form $\forall v(\bigvee_{i=1}^{\ell} \theta_i)$ with $v \in \mathsf{free}(\theta_1) \cap \cdots \cap \mathsf{free}(\theta_\ell)$, the algorithm proceeds dually to the previous case.

□

Proof of Lemma 5.11

Proof. Given a formula ϕ of the described form having $|V| \geqslant 1$ and an elimination ordering $e = ((v_1, \ldots, v_m), E)$ of it, we explain how eliminate the last variable; precisely speaking, we explain how to compute a logically equivalent ψ of the form $\exists\{v_1, \ldots, v_{m-1}\} \bigwedge \psi_j$ and having elimination ordering $((v_1, \ldots, v_{m-1}), E \cap K(\{v_1, \ldots, v_{m-1}\}))$. The desired algorithm iterates this variable elimination.

Let ϕ^{v_m} denote the conjunction of the formulas of the form ϕ_i having $v_m \in \mathsf{free}(\phi_i)$, and let I denote the set containing the indices of the remaining formulas ϕ_i. The formula ψ is defined as $\exists\{v_1, \ldots, v_{m-1}\}$ followed by the conjunction of all formulas in $\Psi = \{\phi_i \mid i \in I\} \cup \{\exists v_m \phi^{v_m}\}$. It is clear that ψ and ϕ are logically equivalent, and we have $|\mathsf{free}(\phi^{v_m})| = 1 + \mathsf{lower\text{-}deg}(e, v_m)$. We verify that $e^\psi = ((v_1, \ldots, v_{m-1}), E \cap K(\{v_1, \ldots, v_{m-1}\}))$ is an elimination ordering of ψ as follows. Since $\mathsf{free}(\psi) = \mathsf{free}(\phi)$, it is clear that the variables in $\mathsf{free}(\psi)$ occur first in the ordering $(v_1, \ldots, v_{m-1})$. To verify that e^ψ is an elimination ordering of the hypergraph named in Definition 5.10, we need to verify that, for each edge f of that hypergraph, one has the containment $K(f) \subseteq E$. For each such edge f, other than the edge $\mathsf{free}(\exists v_m \phi^{v_m})$, this containment holds because it held for the original elimination ordering for ϕ. For the edge $\mathsf{free}(\exists v_m \phi^{v_m})$, the containment holds due to the lower neighbor property and due to v_m being a neighbor of each element of $\mathsf{free}(\phi^{v_m}) \backslash \{v_m\}$ in E.

We now confirm that we can apply the algorithm to ψ to obtain the desired formula ϕ'. We have $\mathsf{lower\text{-}deg}(e^\psi) \leqslant \mathsf{lower\text{-}deg}(e)$ and $\mathsf{width}(\phi^{v_m}) \leqslant \max(\{\mathsf{width}(\phi_i) \mid i \in \underline{n} \backslash I\} \cup \{1 + \mathsf{lower\text{-}deg}(e)\})$. From this, it follows that
$$\max(\{1 + \mathsf{lower\text{-}deg}(e^\psi)\} \cup \{\mathsf{width}(\psi_j) \mid \psi_j \in \Psi\})$$
$$\leqslant \max(\{1 + \mathsf{lower\text{-}deg}(e)\} \cup \{\mathsf{width}(\phi_i) \mid i \in \underline{n}\}).$$
□

Proof of Corollary 6.3

Proof. When Φ has bounded thickness, the claim follows directly from Theorem 6.2 and Proposition 4.6. When Φ does not have bounded thickness, Theorem 6.2 and Proposition 4.7 imply that $\mathsf{param\text{-}MC}[\Phi]$ is either W[1]-hard or co-W[1]-hard, from which the claim follows. □

Proof of Corollary 6.4

Proof. When Φ has bounded thickness, the claim follows directly from Theorem 6.2 and Proposition 4.6. When Φ does not have bounded thickness, Theorem 6.2 and Proposition 4.7 imply that $\mathsf{param\text{-}MC}[\Phi]$ is either non-uniformly W[1]-hard or non-uniformly co-W[1]-hard, so, invoking the fact that nu-FPT is closed under nu-FPT-reductions, the claim follows. □

Discussion: Derivation of the Classification of Prefixed Graphs

Let us say that a sentence is *quantified conjunctive* if conjunction ($\wedge$) is the only connective that occurs therein. A *prefixed graph* is a pair (P, G) where P is a quantifier prefix and G is a graph whose vertices are the variables appearing in P. Let $\mathcal{G}$ be a set of prefixed graphs. Let $\mathrm{QC}(\mathcal{G})$ denote the set that contains a prenex quantified conjunctive sentence $P\phi$ if there exists a prefixed graph $(P, G) \in \mathcal{G}$ such that ϕ is a conjunction of atoms, where the variable set of each atom forms a clique in G. We will assume that no variable occurs more than once in an atom; we make this assumption without loss of interestingness, since given a sentence $\Phi \in \mathrm{QC}(\mathcal{G})$ and a structure $\mathbf{B}$, one can efficiently compute a sentence $\Phi' \in \mathrm{QC}(\mathcal{G})$ and a structure $\mathbf{B}'$ such that (1) each atom of Φ containing more than one variable occurrence is replaced, in Φ', with an atom with the same variables but not having multiple variable occurrences, and (2) $\mathbf{B} \models \Phi$ iff $\mathbf{B}' \models \Phi'$.

The previous work [5] studied the complexity of model checking on sentence sets $\mathrm{QC}(\mathcal{G})$, proving a comprehensive classification. Here, we show how this classification can be readily derived using the main theorem of the present article. This witnesses the strength and generality of our main theorem (Theorem 6.2). We first present two lemmas, then proceed to the derivation.

When $\mathcal{G}$ is a set of prefixed graphs, let $\text{norm-QC}(\mathcal{G})$ denote the subset of $\mathrm{QC}(\mathcal{G})$ that contains a sentence if

- in each atom, the latest occurring variable (in the quantifier prefix) is existentially quantified, and
- it is symbol-loose.

Lemma .6. Let $\mathcal{G}$ be any set of prefixed graphs. The case problems $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$ and $\mathsf{MC}[\text{norm-QC}(\mathcal{G})]$ slice reduce to each other.

Proof. It is straightforward to verify that $\mathsf{MC}[\text{norm-QC}(\mathcal{G})]$ slice reduces to $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$ (by definition, $\text{norm-QC}(\mathcal{G})$ is a subset of $\mathrm{QC}(\mathcal{G})$). We slice reduce from $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$ to $\mathsf{MC}[\text{norm-QC}(\mathcal{G})]$ as follows. Define U to contain a pair (ϕ, ϕ') of prenex quantified conjunctive sentences if they share the same quantifier prefix, ϕ' is symbol-loose, and each atom α of ϕ can be placed in bijective correspondence with an atom α' of ϕ' in such a way that the variables of α' is the set obtained by taking the variables of α and iteratively eliminating the latest occurring variable when it is universally quantified, until the latest occurring variable is existentially quantified. The partial function r is defined in a natural way, namely, such that $r(\phi, \phi', \mathbf{B})$ is a structure $\mathbf{B}'$ having the same universe B as $\mathbf{B}$ and having the following property: for each atom $\alpha = R(u_1, \ldots, u_m)$ of ϕ, it holds that the satisfying assignments for α' over $\mathbf{B}'$ are precisely the satisfying assignments for $\forall D\alpha$ over $\mathbf{B}$, where D denotes the variables in α that do not occur in α'. □

Lemma .7. Let Φ be a set of symbol-loose quantified conjunctive sentences. It holds that $\mathsf{MC}[\Phi]$ and $\mathsf{MC}[\Phi']$ are slice reducible to each other, where Φ' is the graph-like closure of Φ.

Proof. It is straightforward to verify that $\mathsf{MC}[\Phi]$ slice reduces to $\mathsf{MC}[\Phi']$, as $\Phi \subseteq \Phi'$. We thus show that $\mathsf{MC}[\Phi']$ slice reduces to $\mathsf{MC}[\Phi]$. Let U be the set of pairs (ϕ', ϕ) such that ϕ is a symbol-loose quantified conjunctive sentence, and ϕ' is in the graph-like closure of ϕ. We use σ and σ' to denote the signatures of ϕ and ϕ', respectively. Since disjunction and negation do not occur in ϕ, we have that ϕ' is obtained from ϕ via applications of the syntactic transformations (α), (β), and (γ), and replacement. Since each of these syntactic transformations preserves logical equivalence as well as the number of atom occurrences, there is a mapping $S : \sigma \to \sigma'$ such that when each symbol in ϕ is mapped under S, the resulting sentence is logically equivalent to ϕ'. The partial function r is defined by $r(\phi', \phi, \mathbf{B}') = \mathbf{B}$ where for each symbol $R \in \sigma$, the relation $R^{\mathbf{B}}$ is defined as $S(R)^{\mathbf{B}'}$. □

We now explain how to obtain the main dichotomy of the previous work [5], in particular, we show how to classify precisely the sets of prefixed graphs $\mathcal{G}$ such that $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$ is in case-FPT.

First, consider the case where atoms in $\text{norm-QC}(\mathcal{G})$ may have unboundedly many variables. If for each $k \geqslant 1$ there exists an atom in $\text{norm-QC}(\mathcal{G})$ with at least k existentially quantified variables, then there is a direct reduction from case-CLIQUE and one has case-W[1]-hardness of $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$. Otherwise, define F_k to be the sentence $\forall y_1 \ldots \forall y_k \exists x \bigwedge_{i=1}^{k} E_i(y_i, x)$; up to the insertion of additional variables that do not appear in atoms and up to renaming of variables, we have that the sentences F_k are instances of $\text{norm-QC}(\mathcal{G})$. By the above two lemmas, it suffices to prove that the graph-like closure of $\{F_k\}$ is hard. It is readily verified that $\mathrm{lay}^+(F_k)$ is $\forall\{y_1, \ldots, y_k\}(\exists x \bigwedge_{i=1}^{k} E_i(y_i, x))$, where the formula in parentheses is viewed as the single disjunct of a disjunction; we have that $\mathsf{thick}(F_k) = k + 1$, and by the main theorem, we obtain hardness (either case-W[1]-hardness or case-co-W[1]-hardness) of the graph-like closure of $\{F_k\}$.

Next, consider the case where there is a constant upper bound on the number of variables that occur in atoms in $\text{norm-QC}(\mathcal{G})$. By our assumption that no variable occurs more than once in an atom, the set of sentences $\text{norm-QC}(\mathcal{G})$ has bounded arity. In this case, by the two presented lemmas, we have that $\mathsf{MC}[\mathrm{QC}(\mathcal{G})]$ is equivalent, under slice reduction, to $\mathsf{MC}[\Phi']$, where Φ' is the graph-like closure of $\text{norm-QC}(\mathcal{G})$. Since $\text{norm-QC}(\mathcal{G})$ has bounded arity, Φ' does as well. Hence, our main theorem (Theorem 6.2) then can be applied to infer that $\mathsf{MC}[\Phi']$ is either in case-FPT, is case-W[1]-hard, or is case-co-W[1]-hard.

By arguing as in Corollaries 6.3 and 6.4, one can derive dichotomies in the complexity of $\mathsf{param\text{-}MC}[\mathrm{QC}(\mathcal{G})]$.

Proof of Theorem 7.2

Proof. We define a slice reduction (U, r^+) from $Q[\mathsf{closure}_C(S)]$ to $Q[S]$. Set U to be the set containing the pairs (u, u') such that there exists $k \geqslant 1$ and a sequence $u_1, \ldots, u_k$ such that $u = u_1$, $u_k = u'$, and $(u_i, u_{i+1}) \in C$ when $1 \leqslant i < k$. It is straightforwardly verified that U is computably enumerable and that the coverage criterion is satisfied, that is, if $u \in \mathsf{closure}_C(S)$, then there exists $u' \in S$ such that $(u, u') \in U$. Fix A_U to be an algorithm that, given a pair (u, u'), returns a sequence $u = u_1, u_2, \ldots, u_k = u'$ of the just-described form whenever $(u, u') \in U$. Consider the algorithm A_{r^+} that does the following: given a triple (u, u', y), it invokes $A_U(u, u')$; if this computation halts with output $u = u_1, u_2, \ldots, u_k = u'$, the algorithm sets $y_1 = y$; the algorithm computes

$$y_2 = r(u_1, u_2, y_1),\ y_3 = r(u_2, u_3, y_2),\ \ldots,\\ y_k = r(u_{k-1}, u_k, y_{k-1}),$$

and outputs y_k. We define r^+ as the partial mapping computed by this algorithm A_{r^+}. In order to compute each of the strings $y_2, \ldots, y_k$, the algorithm A_{r^+} uses the algorithm A_r for r provided by the definition of an accordion reduction; let f_r and p_r be a function and a polynomial, respectively, such that the running time of A_r, on an input (u, u', y), is bounded above by $f_r(u, u')p_r(|(u, u', y)|)$.

On an input (u, u', y) where r^+ is defined, the running time of A_{r^+} can be bounded above by the running time of $A_U(u, u')$ plus

$$f_r(u_1, u_2)p_r(|(u_1, u_2, y_1)|) + \cdots$$

$$+ f_r(u_{k-1}, u_k)p_r(|(u_{k-1}, u_k, y_{k-1})|).$$

Note that the running time of $A_U(u, u')$ and k are both functions of (u, u'). So, in order to show that the running time of $A_{r^+}(u, u', y)$ is FPT-computable with respect to (π_1, π_2), it suffices to show that, for each $i \geqslant 1$, the term $f_r(u_i, u_{i+1})p_r(|(u_i, u_{i+1}, y_i)|)$ is degree-bounded, when defined (hereon, when discussing degree-boundedness, this is with respect to (π_1, π_2)). So consider such a term; the strings u_i and u_{i+1} are functions of (u, u'), and so by appeal to Proposition 3.11, it suffices to show that the string length $|y_i|$ is degree-bounded. This is clear in the case that $i = 1$, that is, we have that $|y_1|$ is degree-bounded. Let us consider the case that $i > 1$. By the measure-linearity of r, we have that

$$m(u_i, y_i) \leqslant B_{(u_1, u_2)} \cdots B_{(u_{i-1}, u_i)} m(u_1, y_1).$$

By the definition of measure, it follows that

$$m(u_i, y_i) \leqslant B_{(u_1, u_2)} \cdots B_{(u_{i-1}, u_i)} |y_1|.$$

Since the constants $B_{(u_j,u_{j+1})}$ depend only on (u, u'), we obtain that the value $m(u_i, y_i)$ is degree-bounded. Letting f_m and p_m be the function and polynomial (respectively) provided by the definition of measure, we then have $|y_i| \leqslant f_m(u_i)p_m(m(u_i, y_i))$, and conclude that $|y_i|$ is degree-bounded. □

Proof of Lemma 8.2

Proof. Define Φ' to be the set that contains a loose layered sentence if it occurs as a positively combined subformula of a loose sentence in $\mathrm{lay}^+(\{\phi \in \Phi \mid \phi \text{ is loose}\})$. We have that $\mathsf{thick}_l(\Phi')$ is unbounded; this is because, for each sentence $\phi \in \Phi$, if we define ϕ^L to be equal to $\mathrm{org}^+(\phi)$ but with symbols renamed (if necessary) so that ϕ^L is loose, then $\phi^L \in \Phi$ and it is straightforwardly verified that $\mathsf{thick}(\phi) = \mathsf{thick}(\phi^L)$.

We claim that $\mathsf{MC}[\Phi']$ slice reduces to $\mathsf{MC}[\Phi]$. We define a slice reduction as follows. The set U contains a pair (ϕ', ϕ) if ϕ is a loose sentence and ϕ' is a loose layered sentence that is a positively combined subformula of $\mathrm{lay}^+(\phi)$; we have that U is computable. For pairs $(\phi', \phi) \in U$, we set $r(\phi', \phi, \mathbf{B}') = \mathbf{B}$, where $\mathbf{B}$ is defined as follows. For each symbol R (of arity k) that appears in ϕ but not in ϕ', we define $R^{\mathbf{B}} = B^k$ or $R^{\mathbf{B}} = \varnothing$ as appropriate so that $\mathbf{B}' \models \phi'$ if and only if $\mathbf{B} \models \phi$; this is possible since ϕ' is a positively combined subformula of $\mathrm{lay}^+(\phi)$ and because $\mathrm{lay}^+(\phi)$ and ϕ are logically equivalent (by Theorem 5.5).

Define Ψ to be the set that contains a sentence ψ if it can be obtained from a sentence $\phi' \in \Phi'$ by removing all negations that appear immediately in front of atoms. Since each sentence in Φ' is loose and layered, we obtain that each sentence in Ψ is friendly. We give a slice reduction from $\mathsf{MC}[\Psi]$ to $\mathsf{MC}[\Phi']$, as follows. Define U to be the set that contains a pair (ψ, ϕ') if ϕ' is a loose layered sentence and ψ can be obtained from ϕ' by removing all negations that appear in front of atoms. When $(\psi, \phi') \in U$, define $r(\psi, \phi', \mathbf{B}) = \mathbf{B}'$ where $B' = B$ and, for each symbol R of arity k, it holds that $R^{\mathbf{B}'} = B^k \backslash R^{\mathbf{B}}$ when R appears in ϕ' with a negation before it, and $R^{\mathbf{B}'} = R^{\mathbf{B}}$ otherwise. (Note that $B^k \backslash R^{\mathbf{B}}$ can always be computed in polynomial time from $R^{\mathbf{B}}$ due to our assumption of bounded arity.) From the definition of $\mathsf{thick}_l(\cdot)$, we have that $(\psi, \phi') \in U$ implies $\mathsf{thick}_l(\psi) = \mathsf{thick}_l(\phi')$, so $\mathsf{thick}_l(\Psi)$ is unbounded.

By Theorem 4.4, there is a slice reduction from $\mathsf{MC}[\Psi]$ to $\mathsf{MC}[\Phi]$. □