Optimally Rewriting Formulas and Database Queries: A Confluence of Term Rewriting, Structural Decomposition, and Complexity

Hubie Chen
Department of Informatics, King's College London

Stefan Mengel
Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL)

Abstract

A central computational task in database theory, finite model theory, and computer science at large is the evaluation of a first-order sentence on a finite structure. In the context of this task, the *width* of a sentence, defined as the maximum number of free variables over all subformulas, has been established as a crucial measure, where minimizing width of a sentence (while retaining logical equivalence) is considered highly desirable. An undecidability result rules out the possibility of an algorithm that, given a first-order sentence, returns a logically equivalent sentence of minimum width; this result motivates the study of width minimization via syntactic rewriting rules, which is this article's focus. For a number of common rewriting rules (which are known to preserve logical equivalence), including rules that allow for the movement of quantifiers, we present an algorithm that, given a positive first-order sentence ϕ, outputs the minimum-width sentence obtainable from ϕ via application of these rules. We thus obtain a complete algorithmic understanding of width minimization up to the studied rules; this result is the first one—of which we are aware—that establishes this type of understanding in such a general setting. Our result builds on the theory of term rewriting and establishes an interface among this theory, query evaluation, and structural decomposition theory.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Database query processing and optimization (theory)

Keywords and phrases width, query rewriting, structural decomposition, term rewriting

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.16

Funding *Hubie Chen*: acknowledges the support of EPSRC grants EP/V039318/1 and EP/X03190X/1.
Stefan Mengel: partially supported by the ANR project EQUUS ANR-19-CE48-0019.

1 Introduction

The problem of evaluating a first-order sentence on a finite structure is a central computational task in database theory, finite model theory, and indeed computer science at large. In database theory, first-order sentences are viewed as constituting the core of SQL; in parameterized complexity theory, many commonly studied problems can be naturally and directly formulated as cases of this evaluation problem. In the quest to perform this evaluation efficiently, the *width* of a sentence—defined as the maximum number of free variables over all subformulas of the sentence—has become established as a central and crucial measure. A first reason for this is that the natural bottom-up algorithm for sentence evaluation exhibits an exponential dependence on the width [21]; this algorithm runs in polynomial time when restricted to any class of sentences having *bounded width*, meaning that there exists a constant bounding their width. In addition, there are a number of dichotomy theorems [17, 16, 10, 12] showing that, for particular fragments of first-order logic, a class of sentences admits tractable evaluation if and only if the class has bounded width, up to logical equivalence. These theorems are

shown under standard complexity-theoretic assumptions, and concern relational first-order logic, which we focus on in this article.

These results all point to and support the desirability of minimizing the widths of logical sentences. However, an undecidability result [7, Theorem 19] immediately rules out the existence of an algorithm that, given a first-order sentence ϕ, returns a sentence of minimum width among all sentences logically equivalent to ϕ; indeed, this undecidability result rules out an algorithm that performs as described even on positive first-order sentences. Despite this non-computability result, given the importance of width minimization, it is natural to seek computable methods for reformulating sentences so as to reduce their width. In this article, we show how to perform width minimization, in an optimal fashion, via established *syntactic rewriting rules* known to preserve logical equivalence. The study of such rules is strongly relevant:

- In database theory, such rules are studied and applied extensively to reformulate database queries with the aim of allowing efficient evaluation [1, Chapter 6].
- Well-known tractable cases of conjunctive query evaluation admit logical characterizations via such rules, such as those of bounded treewidth [18] and bounded hypertreewidth [15].
- Such rules play a key role in obtaining the tractability results of some of the mentioned dichotomy theorems [10, 12].

This article focuses on a collection of common and well-known rewriting rules, which includes rules that are well-established in database theory [1, Figures 5.1 and 6.2]. Our main result is the presentation of an algorithm that, given a positive first-order sentence ϕ, outputs a minimum-width sentence obtainable from ϕ via application of the considered rules. We thus obtain a complete algorithmic understanding of width minimization up to the studied rules. Our main result is the first result—of which we are aware—to obtain a comprehensive characterization of width up to syntactic rules in the general setting of positive first-order logic.

In order to obtain our main result, we use the theory of term rewriting; in particular, we make use of basic concepts therein such as *termination* and *confluence*. We view this marriage of *term rewriting* and *query rewriting* as a conceptual contribution of our work. We gently augment the basic theory of term rewriting in two ways. First, we define the notion of a *gauged system*, which is an abstract reduction system along with a *gauge*, a function mapping each element of the system to a number; the gauge represents a quantity that one wants to minimize. In this article our focus is on systems whose elements are first-order formulas, and where the gauge is the width. Second, we define a notion of division of a system by an equivalence relation. This notion allows us to consider equivalence classes w.r.t. a subset of the considered rules, in a sense made precise. In particular, we consider equivalence classes w.r.t. a set of rules that correspond to the computation of tree decompositions (see Theorem 23). This will make precise and transparent the role of tree decomposition computations in our algorithm; this also opens up the possibility of applying procedures that approximate treewidth in a black-box fashion, or decomposition methods that are designed for unbounded-arity query classes, such as those associated with hypertree width [14]. Relatedly, we mention here that for any conjunctive query, the minimum width obtainable by the studied rules is equal to the query's treewidth, plus one (see Corollary 24); thus, this measure of minimum width, when looked at for positive first-order logic, constitutes a generalization of treewidth.

All together, the presented framework and theory initiate a novel interface among term rewriting, query rewriting, and structural decomposition theory. In our view, the obtention of our main theorem necessitates the development of novel ideas and techniques that draw upon,

advance, and fuse together these areas. We believe that further development of this interface has the potential to further unite these areas via asking new questions and prompting new techniques—of service to efficient query evaluation. We wish to emphasize, as a conceptual contribution, the marriage of *term rewriting* and *query rewriting*, which is (to our knowledge) novel, despite an enduring overlap in names!

We wish to add two technical remarks. First, although our results are presented for positive first-order logic, our width minimization procedure can straightforwardly be applied to general first-order sentences by first turning it into negation normal form, i.e., pushing all negations down to the atomic level, and then treating negated atoms as if they were atoms. Second, our main result's algorithm straightforwardly gives rise to a fixed-parameter tractability result: on any class of sentences that have bounded width when mapped under our algorithm, we obtain fixed-parameter tractability of evaluating the class, with the sentence as parameter, via invoking our algorithm and then performing the natural bottom-up algorithm for sentence evaluation.

Related work

Some of the previous characterizations and studies of width, for example [10, 12, 8], focused on subfragments of positive first-order logic defined by restricting the permitted quantifiers and connectives; as mentioned, we here consider full positive first-order logic. Width measures for first-order logic have previously been defined [2, 11]. While the measure of [11] makes use of syntactic rewriting rules, and the work [2] studies the relationship of width to syntactic rewriting rules, the present work shows how to optimally minimize formulas up to the set of studied syntactic rewriting rules, a form of result that (to our knowledge) is not entailed by the theory of either of these previous works.[1] In our view, this situation gives our width measure a clear interpretation. This situation also renders some of the previously defined width measures as having, in retrospect, an ad hoc character: as an example, the width measure of [12] has the property that, when a formula has measure w, it is possible to apply rewriting rules to the formula so that it has width w, but the optimal width achievable via these rules is not addressed; the considered rule set from [12] is encompassed by the rule set studied here.

2 Preliminaries

We assume basic familiarity with the syntax and semantics of relational first-order logic, which is our logic of focus. We assume relational structures to have nonempty universes. In building formulas, we assume that conjunction ($\wedge$) and disjunction ($\vee$) are binary. When ψ is a first-order formula, we use $\mathrm{free}(\psi)$ to denote the set of free variables of ψ. By a *pfo-formula*, we refer to a positive first-order formula, i.e., a first-order formula without any negation. We use PFO to denote the class of all pfo-formulas. The *width* of a first-order formula ϕ, denoted by $\mathsf{width}(\phi)$, is defined as the maximum of $|\mathrm{free}(\psi)|$ over all subformulas ψ of ϕ. We will often identify pfo-formulas with their (syntax) trees, that is, trees whose internal nodes are labeled with the operations of first-order logic or labeled with atoms. We also sometimes refer to a node of a syntax tree labeled by an connective $\oplus \in \{\wedge, \vee\}$ simply as a $\oplus$-*node*. We

[1] We also mention that the class of formulas in this article's Example 5 has unbounded width under the width measure of [2] (see their Proposition 3.16), but has bounded width under the width notion of the present article. Thus, insofar as the work [2] exploits syntactic rewriting rules, it does not exploit the ones used in Example 5.

make the convention that in syntax trees quantifiers and the variables that they bind are in the label of a single node.

2.1 Hypergraphs and tree decompositions

A *hypergraph* is a pair (V, E) where V is a set called the *vertex set* of the hypergraph, and E is a subset of the power set of V, that is, each element of E is a subset of V; E is called the *edge set* of the hypergraph. Each element of V is called a *vertex*, and each element of E is called an *edge*. In this work, we only consider finite hypergraphs, so we tacitly assume that V is always a finite set. We sometimes specify a hypergraph simply via its edge set E; we do this with the understanding that the hypergraph's vertex set is $\bigcup_{e \in E} e$. When H is a hypergraph, we sometimes use $V(H)$ and $E(H)$ to denote its vertex set and edge set, respectively. We consider a *graph* to be a hypergraph where every edge has size 2.

A *tree decomposition* of a hypergraph H is a pair $(T, \{B_t\}_{t \in V(T)})$ consisting of a tree T and, for each vertex t of T, a *bag* B_t that is a subset of $V(H)$, such that the following conditions hold:

- (vertex coverage) for each vertex $v \in V(H)$, there exists a vertex $t \in V(T)$ such that $v \in B_t$;
- (edge coverage) for each edge $e \in E(H)$, there exists a vertex $t \in V(T)$ such that $e \subseteq B_t$;
- (connectivity) for each vertex $v \in V(H)$, the set $S_v = \{t \in V(T) \mid v \in B_t\}$ induces a connected subtree of T.

We define the *bagsize* of a tree decomposition $C = (T, \{B_t\}_{t \in V(T)})$, denoted by $\mathsf{bagsize}(C)$, as $\max_{t \in V(T)} |B_t|$, that is, as the maximum size over all bags. The *treewidth* of a hypergraph H is the minimum, over all tree decompositions C of H, of the quantity $\mathsf{bagsize}(C) - 1$. A *minimum width tree decomposition* of a hypergraph H is a tree decomposition C where $\mathsf{bagsize}(C) - 1$ is the treewidth of H.

2.2 Term rewriting

Basics

We here introduce the basic terminology of term rewriting to be used; for the most part, we base our presentation on [3, Chapter 2]. A *system* is a pair $(D, \to)$ where D is a set, and $\to$ is a binary relation on D; the binary relation $\to$ will sometimes be called a *reduction*. Whenever $\to$ is a binary relation, we use $\leftarrow$ to denote its *inverse* (so $a \to b$ if and only if $b \leftarrow a$), and we use $\leftrightarrow$ to denote the union $\to \cup \leftarrow$ which is straightforwardly verified to be a symmetric relation. We use $\overset{*}{\to}$, $\overset{*}{\leftarrow}$, and $\overset{*}{\leftrightarrow}$ to denote the reflexive-transitive closures of $\to$, $\leftarrow$, and $\leftrightarrow$, respectively. Note that $\overset{*}{\leftrightarrow}$ is an equivalence relation.

We will use the following properties of elements of a system. Let $(D, \to)$ be a system, and let $d, e \in D$.

- d is *reducible* if there exists $d' \in D$ such that $d \to d'$.
- d is *in normal form* if it is not reducible.
- e is a *normal form* of d if $d \overset{*}{\to} e$ and e is in normal form.
- d and e are *joinable*, denoted $d \downarrow e$, if there exists an element f such that $d \overset{*}{\to} f \overset{*}{\leftarrow} e$.

Relative to a system $(D, \to)$, an element $d \in D$, and a binary relation $\to'$, we say that $\to'$ is *applicable to* d or *can be applied to* d if there exists an element $e \in D$ such that $d \to' e$.

We next define properties of a system $Y = (D, \to)$; in what follows, d, e, e', d_i, etc. denote elements of D.

- Y is *confluent* if $e \overset{*}{\leftarrow} d \overset{*}{\to} e'$ implies $e \downarrow e'$.

- Y is *locally confluent* if $e \leftarrow d \rightarrow e'$ implies $e \downarrow e'$.
- Y is *terminating* if there is no infinite descending chain $d_0 \rightarrow d_1 \rightarrow \cdots$ of elements in D.
- Y is *convergent* if it is confluent and terminating.

The property of being terminating immediately implies that of being *normalizing*, meaning that each element has a normal form.

We will use the following properties of convergent systems.

▶ **Proposition 1** (see [3], Lemma 2.1.8 and Theorem 2.1.9)**.** *Suppose that $(D, \rightarrow)$ is a convergent system. Then, each element $d \in D$ has a unique normal form. Moreover, for all elements $d, e \in D$, it holds that $d \overset{*}{\leftrightarrow} e$ if and only if they have the same normal form.*

Whenever an element $d \in D$ has a unique normal form, we denote this unique normal form by $d{\downarrow}$. The last part of Proposition 1 can then be formulated as follows: for every convergent system $(D, \rightarrow)$ we have for all elements $d, e \in D$ that $d \overset{*}{\leftrightarrow} e$ if and only if $d{\downarrow} = e{\downarrow}$.

The following lemma, often called *Newman's Lemma*, was first shown in [19] and will be used to establish convergence. See also [3, Lemma 2.7.2] for a proof.

▶ **Lemma 2** ([19])**.** *If a system is locally confluent and terminating, then it is confluent, and hence convergent.*

Gauged systems

We next extend the usual setting of term rewriting by considering systems having a *gauge*, which intuitively is a measure that one desires to minimize by rewriting. To this end, define a *gauged system* to be a triple $(D, \rightarrow, g)$ where $(D, \rightarrow)$ is a system, and $g : D \rightarrow \mathbb{R}$ is a mapping called a *gauge*. A gauged system is *monotone* if for any pair of elements $d, e \in D$ where $d \rightarrow e$, it holds that $g(d) \geq g(e)$, that is, applying the reduction cannot increase the gauge. We apply the terminology of Section 2.2 to gauged systems; for example, we say that a gauged system $(D, \rightarrow, g)$ is *convergent* if the system $(D, \rightarrow)$ is convergent.

▶ **Proposition 3.** *Let $(D, \rightarrow, g)$ be a gauged system that is monotone and convergent. Then, for any elements $d, e \in D$, it holds that $d \overset{*}{\leftrightarrow} e$ implies $g(d{\downarrow}) \leq g(e)$. That is, for any element d, its normal form $d{\downarrow}$ has the minimum gauge among all elements e with $d \overset{*}{\leftrightarrow} e$.*

Proof. First note that, by a simple induction using the monotonicity of g, we have for all elements $e', e'' \in D$ that $e' \overset{*}{\rightarrow} e''$ implies $g(e') \geq g(e'')$. Next, since $(D, \rightarrow)$ is convergent, we have by Proposition 1 that $d{\downarrow} = e{\downarrow}$. But then $g(e) \geq g(e{\downarrow}) = g(d{\downarrow})$. ◀

Systems and division

We here define a notion of dividing systems by equivalence relations. Let D be a set and let $\equiv$ be an equivalence relation on D. Let $D/\equiv$ denote the set containing the $\equiv$-equivalence classes. We define $(D, \rightarrow)/\equiv$ as the system $(D/\equiv, \rightarrow)$, where we extend the definition of $\rightarrow$ to $D/\equiv$ by positing that for $\equiv$-equivalence classes C, C', it holds that $C \rightarrow C'$ if and only if there exist $d \in C$ and $d' \in C'$ such that $d \rightarrow d'$.

Suppose that $G = (D, \rightarrow, g)$ is a gauged system. We extend the definition of g so that, for each set $S \subseteq D$, we have $g(S) = \inf\{g(d) \mid d \in S\}$. We define $G/\equiv$ as the system $(D/\equiv, \rightarrow, g)$.

3 Rewriting rules

We here define well-known transformation rules on first-order formulas, which are all known to preserve logical equivalence. We define these rules with the understanding that they can always be applied to subformulas, so that when ψ_1 is a subformula of ϕ_1 and $\psi_1 \to \psi_2$, we have $\phi_1 \to \phi_2$, where ϕ_2 is obtained from ϕ_1 by replacing the subformula ψ_1 with ψ_2. In the following F_1, F_2, etc. denote formulas. It will be convenient to formalize these rules as rewriting rules in the sense defined before.

The rule $\to_A$ is *associativity* for $\oplus \in \{\wedge, \vee\}$:

$$(F_1 \oplus (F_2 \oplus F_3)) \to_A ((F_1 \oplus F_2) \oplus F_3), \quad ((F_1 \oplus F_2) \oplus F_3) \to_A (F_1 \oplus (F_2 \oplus F_3)).$$

The rule $\to_C$ is *commutativity* for $\oplus \in \{\wedge, \vee\}$:

$$(F_1 \oplus F_2) \to_C (F_2 \oplus F_1).$$

The *pushdown* rule $\to_{P\downarrow}$ applies when F_2 is a formula in which x is not free:

$$\exists x(F_1 \wedge F_2) \to_{P\downarrow} (\exists x F_1) \wedge F_2, \quad \forall x(F_1 \vee F_2) \to_{P\downarrow} (\forall x F_1) \vee F_2.$$

The *pushup* rule $\to_{P\uparrow}$ is defined as the inverse of $\to_{P\downarrow}$.

The *reordering* rule $\to_O$ is defined when $Q \in \{\forall, \exists\}$ is a quantifier: $QxQyF \to_O QyQxF$. The *renaming* rule $\to_N$ allows $QxF \to_N QyF'$ when $y \notin \text{free}(F)$ and F' is derived from F by replacing each free occurrence of x with y.

Collectively, we call the above rules the *tree decomposition rules*, since, as we will see in Section 8, they allow, in a way that we will make precise, optimizing the treewidth of certain subformulas.

The *splitdown* rule distributes quantification over a corresponding connective:

$$\exists x(F_1 \vee F_2) \to_{S\downarrow} (\exists x F_1) \vee (\exists x F_2), \quad \forall x(F_1 \wedge F_2) \to_{S\downarrow} (\forall x F_1) \wedge (\forall x F_2).$$

The *splitup* rule $\to_{S\uparrow}$ is defined as the inverse of $\to_{S\downarrow}$.

The *removal* rule $\to_M$ allows removal of a quantification when no variables are bound to the quantification: when $Q \in \{\forall, \exists\}$, we have $QxF \to_M F$ when F contains no free occurrences of the variable x, that is, when $x \notin \text{free}(F)$.

We tacitly use the straightforwardly verified fact that when the given rules are applied to any pfo-formula, the formula's set of free variables is preserved. We denote these rules via the given subscripts. We use $\mathcal{T}$ to denote the set of all tree decomposition rules, so $\mathcal{T} = \{A, C, P\downarrow, P\uparrow, O, N\}$. We use $\mathcal{A}$ to denote the set of all presented rules, so $\mathcal{A} = \mathcal{T} \cup \{S\downarrow, S\uparrow, M\}$.

When we have a set $\mathcal{R}$ of subscripts representing these rules, we use $\to_{\mathcal{R}}$ to denote the union $\bigcup_{R\in\mathcal{R}} \to_R$. For example, $\to_{\{A,C,O\}}$ denotes $\to_A \cup \to_C \cup \to_O$. In this context, we sometimes denote a set by the string concatenation of its elements, for example, we write $\to_{ACO}$ in place of $\to_{\{A,C,O\}}$. We apply these conventions to $\leftarrow, \leftrightarrow, \overset{*}{\to}, \overset{*}{\leftarrow}, \overset{*}{\leftrightarrow}$, and so forth. When $\mathcal{R}$ is a set of rule subscripts and ϕ is a formula, we use $[\phi]_{\mathcal{R}}$ to denote the $\overset{*}{\leftrightarrow}_{\mathcal{R}}$-equivalence class of ϕ. So, for example, $[\phi]_{\mathcal{T}}$ denotes the $\overset{*}{\leftrightarrow}_{\mathcal{T}}$-equivalence class of ϕ, and $[\phi]_{ACO}$ denotes the $\overset{*}{\leftrightarrow}_{ACO}$-equivalence class of ϕ.

▶ **Example 4.** Let $\phi = \exists x \exists y \exists t(R(x,t) \wedge S(t,y))$; this sentence has width 3. We have

$$\phi \overset{*}{\to}_O \exists t \exists x \exists y(R(x,t) \wedge S(t,y)) \overset{*}{\to}_{P\downarrow} \exists t((\exists x R(x,t)) \wedge (\exists y S(t,y))).$$

The last formula has width 2; the rules we used were in $\mathcal{T}$. It can, however, be verified that, using the rules in $\mathcal{T} \setminus \{O\}$, this formula ϕ cannot be rewritten into a width 2 formula: using these rules, any rewriting will be derivable from ϕ via commutativity and renaming.

▶ **Example 5.** Consider the formulas $(\phi_n)_{n\geq 1}$ defined by $\phi_n = \exists x_1 \dots \exists x_n \forall y (\bigwedge_{i=1}^{n} E_i(x_i, y))$; the formula ϕ_n has width $n+1$. (Although formally we consider conjunction as a binary connective, we allow higher-arity conjunctions due to the presence of the associativity rule.) We have

$$\phi_n \overset{*}{\to}_{S\downarrow,A} \exists x_1 \dots \exists x_n (\bigwedge_{i=1}^{n} \forall y E_i(x_i, y)) \overset{*}{\to}_{P\downarrow,A} \bigwedge_{i=1}^{n} (\exists x_i \forall y E_i(x_i, y)).$$

The last formula has width 2, and hence the rules we consider suffice to convert each formula ϕ_n into a width 2 formula.

Justification of the rule choice

Before stating our main result in the next section, let us take some time to discuss why we have chosen the above rules for study, in this paper. First, as suggested in the article, these rules appear in a number of textbooks and well-known sources. In addition to being in the standard database book [1], one can find some of the crucial ones in [20, page 99], and many of the crucial ones also appear in the Wikipedia entry on first-order logic.[2] We can remark that these rules are generally used when showing that first-order formulas can be rewritten into prenex normal form.

Indeed, apart from distributivity (discussed in the conclusion), we are not aware of any other syntactic rewriting rules (apart from combinations of the rules that we study); in particular, we did not find any others in any of the standard sources that we looked at.

As alluded to in the introduction, the study of many of these rules is strongly established in the research literature. They are studied in articles including [2, 7, 12, 10, 11, 13, 15, 18]. As mentioned, subsets of the rule set are used crucially, in the literature, to obtain the positive results of dichotomy theorems. In particular, there are precise contexts where a subset of the rule set is sufficient to give a tractability result (for example, in [12]). The tractability result of the present article strengthens these previous tractability results since it optimally minimizes width with respect to the considered rules, and thus provides a wider and deeper perspective on these previous tractability results.

Let us emphasize that a subset of the considered rules corresponds to treewidth computation in a very precise sense (see Theorem 23 and Corollary 24). The correspondence between rewriting rules and computation of tree decompositions was also studied, for example, in [18, 13, 15].

4 Main theorem and roadmap

In this section, we first state our main theorem, then we give the intermediate results that we will show in the remainder of the paper to derive it; at the end of the section, we prove the main theorem using the intermediate results. Our main theorem essentially says the following: there is an algorithm that, given a pfo-formula ϕ, computes a formula ϕ' that is derived from ϕ by applying the defined rules $\mathcal{A}$ and that has the minimum width over

[2] https://en.wikipedia.org/wiki/First-order_logic#Provable_identities

formulas derivable by these rules. Moreover, up to computation of minimum width tree decompositions, the algorithm computing ϕ is efficient.

▶ **Theorem 6.** *(Main theorem) There exists an algorithm A that, given a pfo-formula ϕ, computes a minimum width element of $[\phi]_{\mathcal{A}}$. Moreover, with oracle access to an algorithm for computing minimum width tree decompositions, the algorithm A can be implemented in polynomial time.*

Here, we say that an algorithm *computes minimum width tree decompositions* if, when given a hypergraph H, it outputs a minimum width tree decomposition of H.

In order to establish our main theorem, we study and make use of the systems

$$Y = (\mathrm{PFO}, \to_{\{P\downarrow, S\downarrow, M\}}) \,/\, \overset{*}{\leftrightarrow}_{ACO}, \quad Y' = (\mathrm{PFO}, \to_{\{S\downarrow, M\}}) \,/\, \overset{*}{\leftrightarrow}_{\mathcal{T}} .$$

We also make use of the gauged system

$$G' = (\mathrm{PFO}, \to_{\{S\downarrow, M\}}, \mathsf{width}) \,/\, \overset{*}{\leftrightarrow}_{\mathcal{T}},$$

where, as defined before, width is the function that maps a formula to its width. We show that for the system Y, normal forms can be computed in polynomial time, and moreover, that a normal form of Y directly yields a normal form of Y'.

▶ **Theorem 7.** *The system Y is terminating; moreover, there exists a polynomial-time algorithm that, given a pfo-formula ϕ, returns a pfo-formula ϕ^+ such that $[\phi^+]_{ACO}$ is a normal form of $[\phi]_{ACO}$ in the system Y.*

▶ **Theorem 8.** *Suppose that ϕ is a pfo-formula where $[\phi]_{ACO}$ is a normal form of the system Y; then, $[\phi]_{\mathcal{T}}$ is a normal form of the system Y'.*

We then prove that the system Y' is convergent, and that its corresponding gauged system G' is monotone. Together, these results allow us to leverage Proposition 3 and allow us to compute minimum-width equivalence classes in G'.

▶ **Theorem 9.** *The system Y' is convergent.*

▶ **Theorem 10.** *The gauged system G' is monotone.*

The remaining piece needed is to show that minimization can be performed within a $\overset{*}{\leftrightarrow}_{\mathcal{T}}$-equivalence class, which is what the following theorem supplies.

▶ **Theorem 11.** *There exists an algorithm that, given a pfo-formula θ, outputs a pfo-formula θ^+ having minimum width among all formulas in $[\theta]_{\mathcal{T}}$. With oracle access to an algorithm for computing minimum width tree decompositions, this algorithm can be implemented in polynomial time.*

Proof of the main theorem—Theorem 6. Given a pfo-formula ϕ, the algorithm first applies the algorithm of Theorem 7 to obtain a pfo-formula θ where $[\theta]_{ACO}$ is a normal form of $[\phi]_{ACO}$ in Y, and then applies the algorithm of Theorem 11 to obtain a pfo-formula θ^+ having minimum width among the formulas in $[\theta]_{\mathcal{T}}$; θ^+ is the output of the algorithm.

We justify this algorithm's correctness as follows. As $[\theta]_{ACO}$ is a normal form of $[\phi]_{ACO}$ in Y, we have $[\phi]_{ACO} \overset{*}{\to}_{\{P\downarrow, S\downarrow, M\}} [\theta]_{ACO}$, and thus $[\phi]_{\mathcal{T}} \overset{*}{\to}_{\{S\downarrow, M\}} [\theta]_{\mathcal{T}}$, because all applications of $P\downarrow$ can be simulated by choosing a representative in the equivalence class w.r.t. $\mathcal{T}$. It follows from Theorem 8 that $[\theta]_{\mathcal{T}}$ is a normal form of $[\phi]_{\mathcal{T}}$ in Y'. From Theorems 9 and 10, we have that the gauged system G' is convergent and monotone, so by Proposition 3, we have that, in the gauged system G', the element $[\theta]_{\mathcal{T}}$ has the minimum gauge among all elements that are $\overset{*}{\leftrightarrow}_{\{S\downarrow, M\}}$-related to $[\phi]_{\mathcal{T}}$. Thus, a minimum width element in $[\theta]_{\mathcal{T}}$ is a minimum width element in $[\phi]_{\mathcal{T}\cup\{S\downarrow, S\uparrow, M\}}$. ◀

5 Formulas

In this section, we define a few types of formulas to be used in our development, and show some basic properties thereof.

5.1 Holey formulas

A *holey formula* is intuitively defined similarly to a formula, but in lieu of atoms, there are placeholders where further formulas can be attached; these placeholders are represented by natural numbers. Formally, a *holey pfo-formula* is a formula built as follows: each natural number $i \geq 1$ is a holey pfo-formula, and is referred to as a *hole*; when ϕ and ϕ' are holey pfo-formulas, so are $\phi \wedge \phi'$ and $\phi \vee \phi'$; and, when ϕ is a holey pfo-formula and x is a variable, so are $\exists x\phi$ and $\forall x\phi$. We require that for each holey pfo-formula ϕ, no natural number occurs more than once. We say that a holey pfo-formula is *atomic* if it is equal to a natural number (equivalently, if it contains no connectives nor quantifiers). A *holey* $\{\exists, \wedge\}$*-formula* is defined as a holey pfo-formula where, apart from atoms, only the formation rules involving existential quantification and conjunction are permitted. A *holey* $\{\forall, \vee\}$*-formula* is defined dually.

When ϕ is a holey formula with holes among $1, \ldots, k$ and we have that $\psi_1, \ldots, \psi_k$ are formulas, we use $\phi[\![\psi_1, \ldots, \psi_k]\!]$ to denote the formula obtained from ϕ by substituting, for each $i = 1, \ldots, k$, the formula ψ_i in place of i.

▶ **Example 12.** Consider the holey pfo-formula $\phi = 2 \wedge 1$, and the formulas $\psi_1 = S(x) \vee S(y)$, $\psi_2 = R(x, z)$. We have $\phi[\![\psi_1, \psi_2]\!] = R(x, z) \wedge (S(x) \vee S(y))$.

In the sequel, we will speak of applying rewriting rules (generally excluding the rule N) to holey formulas. In order to speak of the applicability of rules such as the pushdown and pushup rules, we need to associate a set of free variables to each subformula of a holey formula. We define an *association* for a holey formula ϕ to be a partial mapping a defined on the natural numbers that is defined on each hole in ϕ, and where, for each number i on which a is defined, it holds that $a(i)$ is a set of variables. With an association a for a holey formula ϕ, we can naturally define a set of free variables on each subformula of ϕ, by considering $\mathrm{free}(i) = a(i)$ for each i on which a is defined, and then using the usual inductive definition of $\mathrm{free}(\cdot)$.

5.2 Standardized formulas

Define a *standardized* formula to be a formula ϕ where, for each occurrence Qx of quantification, the following hold: (1) x is not quantified elsewhere, that is, for any other occurrence $Q'x'$ of quantification, $x \neq x'$ holds; (2) $x \notin \mathrm{free}(\phi)$. Intuitively, a standardized formula is one where there is no name clash between a quantified variable x and other variable occurrences in the formula. It is straightforward to verify that a subformula of a standardized formula is also standardized.

It is straightforward to verify that every first-order formula can be standardized by repeatedly applying the renaming rule. We will use the following formalization of this observation which in effect asserts that in terms of applying the studied rules to reduce width, one may always assume that variable renaming that leads to a standardized formula is always performed upfront.

▶ **Proposition 13.** *Let ϕ be a pfo-formula or a holey pfo-formula along with an association a. Suppose $\phi = \phi_0$ and $\phi_0 \to_{A_1} \phi_1 \cdots \to_{A_t} \phi_t$ where each A_i is in $\mathcal{T} \cup \{S\downarrow, M\}$. Let $A'_1, \ldots, A'_s$ be the sequence obtained from $A_1, \ldots, A_t$ by removing instances of N. Then, there exists a*

standardized formula ϕ'_0 with the following property: there exist formulas $\phi'_1, \ldots, \phi'_s$ where $\phi \xrightarrow{*}_N \phi'_0 \to_{A'_1} \phi'_1 \cdots \to_{A'_s} \phi'_s \xrightarrow{*}_N \phi_t$. Indeed, for any standardized formula ϕ'_0 with $\phi \xrightarrow{*}_N \phi'_0$, the given property holds.

5.3 Organized formulas

We next define and study *organized formulas*, which, intuitively speaking, are pfo-formulas that are stratified into regions, based on the quantifiers and connectives. The first main property we show is that each non-atomic pfo-formula can be viewed as an organized formula.

We define $\{\exists, \wedge\}$-*organized* formulas and $\{\forall, \vee\}$-*organized* formulas by mutual induction, as follows.

- When ϕ is a non-atomic holey $\{\exists, \wedge\}$-formula and each of $\psi_1, \ldots, \psi_k$ is an atom or a $\{\forall, \vee\}$-organized formula, then $\phi[\![\psi_1, \ldots, \psi_k]\!]$ is an $\{\exists, \wedge\}$-organized formula.
- When ϕ is a non-atomic holey $\{\forall, \vee\}$-formula and each of $\psi_1, \ldots, \psi_k$ is an atom or an $\{\exists, \wedge\}$-organized formula, then $\phi[\![\psi_1, \ldots, \psi_k]\!]$ is a $\{\forall, \vee\}$-organized formula.

An *organized formula* is defined to be a formula that is either an $\{\exists, \wedge\}$-organized formula or a $\{\forall, \vee\}$-organized formula.

We call the formula ϕ in the above definition a *region* of the organized formula. Thus, every organized formula θ can be decomposed into atoms and different regions which are maximal holey $\{\exists, \wedge\}$- and $\{\forall, \vee\}$-subformulas of θ. Define the *top operation* of a pfo-formula to be the label of the root of the formula when interpreting it as a tree.

▶ **Proposition 14.** *Each non-atomic pfo-formula θ is an organized formula. More precisely, each non-atomic pfo-formula θ whose top operation is $\exists$ or $\wedge$ is an $\{\exists, \wedge\}$-organized formula. Each non-atomic pfo-formula θ whose top operation is $\forall$ or $\vee$ is an $\{\forall, \vee\}$-organized formula.*

When the rules in $\mathcal{T} \setminus \{N\}$ are applied to organized formulas, they act on regions independently, in the following formal sense.

▶ **Proposition 15.** *Suppose that $\Phi = \phi[\![\psi_1, \ldots, \psi_k]\!]$ is an organized formula. Then, each formula in $[\Phi]_{\mathcal{T}\setminus\{N\}}$ has the form $\phi'[\![\psi'_1, \ldots, \psi'_k]\!]$, where $\phi' \in [\phi]_{\mathcal{T}\setminus\{N\}}$ and $\psi'_1 \in [\psi_1]_{\mathcal{T}\setminus\{N\}}, \ldots, \psi'_k \in [\psi_k]_{\mathcal{T}\setminus\{N\}}$. Here, we understand the $\mathcal{T} \setminus \{N\}$-rules to be applied to ϕ and holey pfo-formulas under the association $i \mapsto \mathrm{free}(\psi_i)$ defined on each $i = 1, \ldots, k$.*

6 Rule applicability

In the remainder of the paper, it will be crucial to have an understanding for when the rules of the systems Y and Y' can be applied. To this end, in this section, we study the structure of formulas that allow their application. This will in particular also lead to the normal-form result of Theorem 8.

We start with the applicability of the removal rule $\to_M$. Remember that we say that a rule $\to$ can be applied to a set Φ of formulas if and only if there is a formula $\phi \in \Phi$ on which $\to$ is applicable.

▶ **Lemma 16.** *Let θ be a pfo-formula. Then the following statements are equivalent:*

- *$\to_M$ can be applied to θ.*
- *$\to_M$ can be applied to $[\theta]_{ACO}$.*
- *$\to_M$ can be applied to $[\theta]_{\mathcal{T}}$.*

Proof sketch. The proof is based on the simple observation that a quantifier does not bind a variable in θ if and only if it does not bind a variable in any representative in $[\theta]_{ACO}$ and $[\theta]_{\mathcal{T}}$. ◀

It will be useful to consider (sets of) formulas in which the pushdown operation has been applied exhaustively, in particular, to understand normal forms of the system Y. To this end, we introduce the following definitions. We say that a pfo-formula ϕ is *pushed-down* if $\to_{P\downarrow}$ cannot be applied to ϕ; we say that a set Φ of pfo-formulas is *pushed-down* if each formula $\phi \in \Phi$ is pushed-down.

We say that a pfo-formula is a *k-fold conjunction* if, up to associativity, it can be written in the form $\psi_1 \wedge \cdots \wedge \psi_k$. We can formalize this as follows. For each pfo-formula θ, define the multiset conjuncts(θ) inductively, as follows. If θ is an atom, or begins with disjunction or quantification, define conjuncts(θ) $= \{\theta\}$, where θ has multiplicity 1. When θ has the form $\theta_1 \wedge \theta_2$, define conjuncts(θ) as the multiset union conjuncts(θ_1) $\cup$ conjuncts(θ_2). We say that θ is a *k-fold conjunction* when $|\text{conjuncts}(\theta)| \geq k$. We define *$k$-fold disjunctions* analogously.

The notion of k-fold conjunction will be useful in the remainder as it allows us to understand when we can apply the splitdown rule on quantifiers. The following lemma tells us that the existence of k-fold conjunctions in our equivalence classes with respect to $\{A, C, O\}$ and $\mathcal{T}$ can be decided by looking at any pushed-down representative.

▶ **Lemma 17.** *Suppose that θ is a pfo-formula with $[\theta]_{ACO}$ pushed-down, and let $k \geq 1$. Then, there exists a k-fold conjunction in $[\theta]_{\mathcal{T}}$ if and only if θ is a k-fold conjunction. Similarly, there exists a k-fold disjunction in $[\theta]_{\mathcal{T}}$ if and only if θ is a k-fold disjunction.*

As a consequence of Lemma 17, we get that for pushed-down formulas there is a result similar to Lemma 16 that characterizes when the pushdown rule can be applied.

▶ **Lemma 18.** *Let θ be a pfo-formula where $[\theta]_{ACO}$ is pushed-down. Then the following are equivalent:*

- *$\to_{S\downarrow}$ can be applied to θ.*
- *$\to_{S\downarrow}$ can be applied to $[\theta]_{ACO}$.*
- *$\to_{S\downarrow}$ can be applied to $[\theta]_{\mathcal{T}}$.*

Proof. The implications from the first point to the second and from the second to the third are again directly clear as in the proof of Lemma 16.

So assume now that $\to_{S\downarrow}$ can be applied to $[\theta]_{\mathcal{T}}$. Then, by definition, there exists a formula $\theta^+ \in [\theta]_{\mathcal{T}}$ to which $\to_{S\downarrow}$ can be applied. Since applicability of $\to_{S\downarrow}$ to a formula and whether or not $[\theta]_{ACO}$ is pushed-down are preserved by variable renaming, we may assume by Proposition 13 that θ is standardized, and that $\theta^+ \in [\theta]_{\mathcal{T}\setminus\{N\}}$. Clearly, the formulas θ and θ^+ are non-atomic. Observe that when $\to_{S\downarrow}$ is applicable to an organized formula, the quantification must come from a holey $\{\exists, \wedge\}$-formula and the connective must come from a holey $\{\forall, \vee\}$-formula, or vice-versa.

By appeal to the decomposition of each non-atomic pfo-formula into an organized formula (Proposition 14), and Proposition 15, there exist organized formulas $\phi[\![\psi_1, \ldots, \psi_k]\!]$, $\phi^+[\![\psi_1^+, \ldots, \psi_k^+]\!]$ that are subformulas of θ and θ^+, respectively, where (1) $\phi \overset{*}{\leftrightarrow}_{\mathcal{T}\setminus\{N\}} \phi^+$ and $\psi_1 \overset{*}{\leftrightarrow}_{\mathcal{T}\setminus\{N\}} \psi_1^+, \ldots, \psi_k \overset{*}{\leftrightarrow}_{\mathcal{T}\setminus\{N\}} \psi_k^+$; and (2) $\to_{S\downarrow}$ can be applied to an instance of quantification in ϕ^+ and a connective in a formula ψ_ℓ^+.

We assume up to duality that ϕ^+ is a holey $\{\forall, \vee\}$-formula and that ψ_ℓ^+ is a $\{\exists, \wedge\}$-organized formula. Viewing ϕ^+ and ψ_ℓ^+ as trees, we have that ℓ occurs as a leaf in ϕ^+, that the parent of this leaf in ϕ^+ is a universal quantification $\forall x$, and that ψ_ℓ^+ has conjunction ($\wedge$) at its root. Thus ψ_ℓ^+ is a 2-fold conjunction; by Lemma 17, it follows that ψ_ℓ is a 2-fold conjunction. Let B be the set of holes occurring in ϕ; since $\phi \overset{*}{\leftrightarrow}_{\mathcal{T}\setminus\{N\}} \phi^+$, we have that B is also the set of holes occurring in ϕ^+. We have free(ψ_i) = free(ψ_i^+) for each $i = 1, \ldots, k$. In applying rules in $\mathcal{T} \setminus \{N\}$ to obtain ϕ from ϕ^+, which of the free occurrences of x from the

formulas ψ_i ($i \in B$) are bound to the mentioned quantification $\forall x$, is preserved. Thus, among the formulas ψ_i ($i \in B$), only ψ_ℓ can have a free occurrence bound to the corresponding quantification $\forall x$ in ϕ. It thus follows that, in ϕ viewed as a tree, there is no instance of disjunction between $\forall x$ and ℓ, for if there were, by quantifier reordering (applying $\to_O$), $\forall x$ could be moved above a disjunction, and then $\to_{P\downarrow}$ would be applicable (contradicting that $[\theta]_{ACO}$ is pushed-down). Thus, in ϕ viewed as a tree, the parent of ℓ is a universal quantification, and so $\to_{S\downarrow}$ can be applied to θ. ◀

With the insights on rule applicability from above, we can now show that there is a polynomial time algorithm that decides if rules from the system $Y = (\mathrm{PFO}, \to_{\{P\downarrow,S\downarrow,M\}}) \,/\, \overset{*}{\leftrightarrow}_{ACO}$ can be applied to an equivalence class with respect to ACO. This will be a building block in the proof of Theorem 7 that shows that we can compute normal forms in the system Y in polynomial time.

▶ **Lemma 19.** *There is a polynomial time algorithm that, given a pfo-formula θ, decides if there is a formula $\theta' \in [\theta]_{ACO}$ such that a rule of Y can be applied to it. If so, it computes such a θ' and a formula θ'' obtainable by applying a rule of Y to θ'.*

Proof. We show this for the three rules in Y individually, starting with $\to_M$, then $\to_{P\downarrow}$ and finally $\to_{S\downarrow}$.

For $\to_M$, by Lemma 16, a quantifier can be deleted in θ if and only if one can be deleted from any $\theta' \in [\theta]_{ACO}$; we thus directly get a polynomial-time algorithm for $\to_M$.

Now assume that $\to_M$ cannot be applied, so every quantifier binds a variable in at least one atom in its subformula in θ. Fix one quantifier v in θ, say that quantifier is universal; the other case is totally analogous. Let θ_v denote the subformula of θ rooted in v. We assume that the quantifiers are checked inductively in a bottom-up fashion, so all quantifiers in strict subformulas of θ_v have been checked before dealing with v. So we cannot apply $\to_{P\downarrow}$ on any of them for any $\theta' \in [\theta]_{ACO}$. By Proposition 14, we have that the subformula θ_v is organized. Let ψ be the region of θ_v that v is applied on, i.e., the largest holey $\{\forall, \vee\}$-subformula in θ containing the root of the subformula that v is applied on. If ψ only has one hole, then clearly we cannot apply $\to_{P\downarrow}$ on v in θ as then ψ does not contain any $\vee$-operation on which we could apply the pushdown. This is also true for all formulas $\theta' \in [\theta]_{ACO}$ since the corresponding holey formula ψ' in θ' also contains no $\vee$-operation, because the rules in $\to_{ACO}$ cannot move them over other operators. So in particular, if we want to apply $\to_{P\downarrow}$ to v, then ψ must contain a $\vee$-operation. After potentially applying $\to_O$ several times, we may assume that the quantification v in θ is applied on a disjunction. Then we can write ψ as an r-fold disjunction

$$\psi = \bigvee_{i=1}^{r} \psi_i, \tag{1}$$

for some integer r where the ψ_i are either holes or have a universal quantifier on top. Now if there is $i^* \in [r]$ such that ψ_{i^*} does not contain x in any of its holes, we can use $\to_A$ to rewrite $\psi = \psi_{i^*} \vee \bigvee_{i \in [r] \setminus \{i^*\}} \psi_i$ which gives us an $\vee$-operation to which we can apply $\to_{P\downarrow}$ for v. On the other hand, if all ϕ_i contain x in one of their holes, then this is the case for all ψ' that we get for θ', because $\to_{ACO}$ does not allow exchanging the positions of $\vee$ and any other operators. Thus, the representation (1) for all ψ' is the same as for ψ up to fact that $\to_{ACO}$ might have been used on the disjuncts ψ_i. But in that case, in no ψ' and thus in no θ' there is a $\vee$-operation that v could be pushed over. This directly yields a polynomial time algorithm for this case.

Finally, assume that $\to_M$ and $\to_{P\downarrow}$ can both not be applied. Then in particular θ is pushed-down. Then, by Lemma 18, we get that $\to_{S\downarrow}$ can be applied to $[\theta]_{ACO}$ if and only if it can be applied to θ which directly yields a polynomial time algorithm.

Overall, we have polynomial time algorithms for all three rules of Y. Observing that if any rule can be applied, we can compute θ' and the result θ'' of the rule application efficiently, completes the proof. ◀

Finally, we give the proof of the normal form Theorem 8 which we restate for the convenience of the reader. Remember that $Y' = (\text{PFO}, \to_{\{S\downarrow,M\}}) \,/\, \overset{*}{\leftrightarrow}_{\mathcal{T}}$.

▶ **Theorem 8.** *Suppose that ϕ is a pfo-formula where $[\phi]_{ACO}$ is a normal form of the system Y; then, $[\phi]_{\mathcal{T}}$ is a normal form of the system Y'.*

Proof. Suppose that $[\phi]_{ACO}$ is a normal form of the system Y. Then, we have that $[\phi]_{ACO}$ is pushed-down, since $P\downarrow$ is a rule in Y. Since $[\phi]_{ACO}$ is a normal form of Y, neither of the rules $P\downarrow$, M can be applied to $[\phi]_{ACO}$, implying by Lemmas 16 and 18 that neither of these two rules can be applied to $[\phi]_{\mathcal{T}}$. Thus $[\phi]_{\mathcal{T}}$ is a normal form of Y'. ◀

7 Termination and confluence

In this section, we will show that both systems Y and Y' are terminating. With Lemma 19 from the last section, this will yield Theorem 7, showing that we can efficiently compute normal forms for the system Y. For Y', we will also show that it is locally confluent which then implies Theorem 9, the convergence of Y'.

We will start with termination for the system Y. Let us for every pfo-formula ϕ denote by $|\phi|$ the number of nodes in the syntax tree of ϕ.

▶ **Lemma 20.** *For every pfo-formula ϕ, any reduction chain in the system Y starting in $[\phi]_{ACO}$ has length at most $|\phi|^3$.*

Proof. Consider a pfo-formula ϕ. We will show that any chain of $\to_{P\downarrow,S\downarrow,M}$-steps starting in $[\phi]_{ACO}$ is upper bounded by $|\phi|^3$. To this end, we define a potential function p on all pfo-formulas as follows: let F be a subformula of ϕ whose root is labeled by a quantifier. Then the local potential $\bar{p}(F)$ of F is defined as a^2 where a is the number of atoms in F. Note that $\bar{p}(F)$ is positive. Then the potential of ϕ is the sum of the local potentials of all subformulas that have a quantifier labeling their root.

We claim that applying any operation in $\to_{P\downarrow,S\downarrow,M}$ decreases the potential. So let ϕ' be obtained from ϕ by applying one reduction step. We consider the different possible cases:

- $\to_{\mathbf{P}\downarrow}$: Let $F = \exists x(F_1 \wedge F_2)$ be a sub-formula of ϕ such that in F_2 the variable x is not free, and consider the application $\exists x(F_1 \wedge F_2) \to_{P\downarrow} \underbrace{(\exists x F_1) \wedge F_2}_{:=F'}$. The only change in the potential of ϕ when applying this rule is the change from $\bar{p}(F)$ to $\bar{p}(F')$, since no other sub-formulas of ϕ change. But F_1 contains fewer atoms than $F_1 \wedge F_2$, so $\bar{p}(F) > \bar{p}(F')$, so the potential of ϕ decreases. All other cases for $\to P\downarrow$ follow analogously.
- $\to_{\mathbf{S}\downarrow}$: Let $F = \forall x(F_1 \wedge F_2)$ be a sub-formula of ϕ and consider the application $\forall x(F_1 \wedge F_2) \to_{S\downarrow} (\forall x F_1) \wedge (\forall x F_2)$. Then this application changes the potential p of ϕ by $\bar{p}(\forall x F_1) + \bar{p}(\forall x F_2) - \bar{p}(F)$. Let a_1 be the number of atoms in F_1 and a_2 the number of atoms in F_2, then

$$\bar{p}(\forall x F_1) + \bar{p}(\forall x F_2) = a_1^2 + a_2^2 < (a_1 + a_2)^2 = \bar{p}(F),$$

so the potential of F decreases. All other cases for $\to_{S\downarrow}$ follow analogously.

- $\to_{\mathbf{M}}$: In the definition of $p(F)$, we lose one positive summand whenever applying $\to_M$ without changing any of the other summands, so the potential decreases.

So in any case, whenever applying one of the rules on ϕ, the potential p decreases.

Now consider the equivalence classes of $PFO \,/\, \overset{*}{\leftrightarrow}_{ACO}$. Note that applying $\to_A$, $\to_C$ or $\to_O$ does not change the potential of any formula, so we can define the potential $[p]$ of every equivalence class $[\phi]_{ACO}$ by $[p]\,([\phi]_{ACO}) := p(\phi)$. Now, whenever applying a rule of the system to an equivalence class $[\phi]_{ACO}$, the potential $[p]([\phi]_{ACO})$ decreases, because, as we have seen before, it decreases for any $\phi' \in [\phi]_{ACO}$.

The potential is bounded by $[p]([\phi]_{ACO}) \leq |\phi|a^2 \leq |\phi|^3$ where $|\phi|$ is the length of ϕ and a the number of atoms. Since the potential is a positive integer, the longest descending chain starting in $[\phi]_{ACO}$ is thus of length $|\phi|^3$, so any chain in the system has polynomially bounded length. ◀

The proof of Theorem 7 follows from this lemma straightforwardly.

We next turn to the termination of Y', which will be used to show Theorem 9 with Lemma 2.

▶ **Lemma 21.** *The system Y' is terminating.*

Proof sketch. The proof follows the same idea as that of Lemma 20: we introduce a potential for every pfo-formula and show that it is the same for all representatives of an equivalence class. Then we show that applying rules from the system Y' decreases the potential of any formula. Unfortunately, the setting is slightly more complicated such that the definition of the potential is more involved than for Lemma 20. ◀

As the second ingredient for the proof of confluence for Y' with the help of Lemma 2, we now show local confluence of Y'.

▶ **Lemma 22.** *The system Y' is locally confluent.*

We now apply Lemma 2 using Lemma 21 and Lemma 22, to directly get Theorem 9.

8 Between rewriting and tree decompositions

The aim of this section is to link tree decompositions and formula width. We will then use our development to prove Theorem 11. In what follows, we focus on $\{\exists, \wedge\}$-formulas, but our results apply to the corresponding dual formulas, namely, $\{\forall, \vee\}$-formulas.

Let ϕ be for now a standardized holey $\{\exists, \wedge\}$-formula, and let a be an association for ϕ. We define the hypergraph of (ϕ, a) as the hypergraph whose vertex set is $\bigcup_i a(i)$, where the union is over all holes i appearing in ϕ, and whose edge set is $\{a(i) \mid i \text{ appears in } \phi\} \cup \{\text{free}(\phi)\}$; that is, this hypergraph has an edge corresponding to each hole of ϕ, and an edge made of the free variables of ϕ.

The following theorem identifies a correspondence between the rules in $\mathcal{T} \setminus \{N\}$ and the computation of minimum tree decompositions. A related result appears as [13, Theorem 7].[3]

▶ **Theorem 23.** *Let ϕ be a standardized holey $\{\exists, \wedge\}$-formula, and let a be an association for ϕ. Let H be the hypergraph of (ϕ, a). It holds that* $\mathsf{width}([\phi]_{\mathcal{T}\setminus\{N\}}) = \mathsf{tw}(H) + 1$. *Moreover, there exists a polynomial-time algorithm that, given the pair ϕ, a and a tree decomposition C*

[3] Let us remark that, in the context of the present theorem, the presence of the rewriting rule O is crucial: Example 4 shows that absence of this rule affects the minimum width achieveable.

of H, outputs a formula $\phi' \in [\phi]_{\mathcal{T}\setminus\{N\}}$ where $\mathsf{width}(\phi') \leq \mathsf{bagsize}(C)$; when C is a minimum width tree decomposition of H, it holds that $\mathsf{width}(\phi') = \mathsf{bagsize}(C)$.

The following is a consequence of Theorem 23, and Proposition 13.

▶ **Corollary 24.** *Let ϕ be a standardized $\{\exists, \wedge\}$-sentence, and let H be the hypergraph of ϕ (defined as having a hyperedge $\{v_1, \ldots, v_k\}$ for each atom $R(v_1, \ldots, v_k)$ of ϕ). It holds that*

$$\mathsf{width}([\phi]_{\mathcal{A}}) = \mathsf{width}([\phi]_{\mathcal{T}}) = \mathsf{width}([\phi]_{\mathcal{T}\setminus\{N\}}) = \mathsf{tw}(H) + 1.$$

Let us briefly explain how to employ Theorem 23 to prove Theorem 11. The proof relies on the fact that, via Proposition 15, we can consider the regions of the formula θ independently. Moreover, Theorem 23 allows us to minimize the width of the individual regions.

9 Monotonicity

In this section, we prove monotonicity of the gauged system G'. We begin by proving a theorem concerning the width, up to rules in $\mathcal{T} \setminus \{N\}$, of holey $\{\exists, \wedge\}$-formulas that are conjunctive. This will allow us to understand how to use these rules to minimize a formula that permits application of the splitdown rule.

▶ **Theorem 25.** *Let θ be a standardized holey $\{\exists, \wedge\}$-formula having the form $\phi \wedge \phi'$, and let a be an association for θ. It holds that $\mathsf{width}([\phi \wedge \phi']_{\mathcal{T}\setminus\{N\}})$ is equal to*

$$\max(\mathsf{width}([\phi]_{\mathcal{T}\setminus\{N\}}), \mathsf{width}([\phi']_{\mathcal{T}\setminus\{N\}}), |\mathrm{free}(\phi) \cup \mathrm{free}(\phi')|).$$

Theorem 25 is used to show Theorem 10.

10 Conclusion

We have seen that one can optimally minimize the width of positive first-order formulas with respect to the application of syntactic rules. The algorithm we have given runs in polynomial time, up to the use of an algorithm computing minimum tree decompositions of hypergraphs. While the computation of such decompositions is known to be NP-hard, there exist FPT-algorithms parameterized by the treewidth (see for example [6, 5]) and it follows directly that our width minimization algorithm runs in FPT-time parameterized by the optimal width.

From our techniques, it can be seen that equivalence up to the syntactical rules that we study can also be checked for pfo-formulas: given two pfo-formulas ϕ_1, ϕ_2, compute their normal forms with the algorithm of Theorem 11. Then, ϕ_1 and ϕ_2 are equivalent if there is an isomorphism of the structure of their regions—which is easy to check since the regions are organized in a tree shape—and the regions mapped onto each other by this isomorphism are equivalent. The latter can be verified by standard techniques for conjunctive queries [9] which yields the algorithm for checking equivalence of pfo-formulas under our syntactical rules and, in the positive case, also allows extracting a sequence of rule applications that transforms ϕ_1 to ϕ_2.

We close by discussing an open issue: it would be interesting to extend the set of rules we allow in the width minimization. A particularly natural addition would be the distributive rules stating that $\wedge$ distributes over $\vee$, and vice-versa. Note that allowing this operation would necessarily change the form of the results we could hope for: it is known that there are formulas whose width minimization leads to an unavoidable exponential blow-up in the

formula size [4], and it is verifiable that this is also true when only allowing syntactical rewriting rules including distributivity. Note that this is very different from our setting where the rewriting rules that we consider may only increase the formula size by adding a polynomial number of quantifiers by the splitdown rule. Thus, when adding distributivity to our rule set, we cannot hope for polynomial time algorithms up to treewidth computation, as we showed in the present article. Still, it would be interesting to understand how distributivity changes the rewriting process beyond this and if there are algorithms that run polynomially, say, in the input and output size.

Acknowledgements

The authors thank Carsten Fuhs, Moritz Müller, and Riccardo Treglia for useful feedback and pointers.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 Isolde Adler and Mark Weyer. Tree-width for first order formulae. *Log. Methods Comput. Sci.*, 8(1), 2012.

3 Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

4 Christoph Berkholz and Hubie Chen. Compiling existential positive queries to bounded-variable fragments. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 353–364. ACM, 2019. doi:10.1145/3294052.3319693.

5 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

6 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An o(cˆk n) 5-approximation algorithm for treewidth. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 499–508. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.60.

7 Simone Bova and Hubie Chen. The complexity of width minimization for existential positive queries. In Nicole Schweikardt, Vassilis Christophides, and Vincent Leroy, editors, *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 235–244. OpenProceedings.org, 2014.

8 Simone Bova and Hubie Chen. How many variables are needed to express an existential positive query? *Theory Comput. Syst.*, 63(7):1573–1594, 2019.

9 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90. ACM, 1977. doi:10.1145/800105.803397.

10 Hubie Chen. On the complexity of existential positive queries. *ACM Trans. Comput. Log.*, 15(1):9:1–9:20, 2014.

11 Hubie Chen. The tractability frontier of graph-like first-order query sets. *J. ACM*, 64(4):26:1–26:29, 2017.

12 Hubie Chen and Víctor Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. *SIAM J. Comput.*, 45(6):2066–2086, 2016.

13 Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In Pascal Van Hentenryck, editor, *Principles and Practice*

of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002. doi:10.1007/3-540-46135-3_21.

14 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: A survey. In Jirí Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, volume 2136 of *Lecture Notes in Computer Science*, pages 37–57. Springer, 2001.

15 Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66(4):775–808, 2003.

16 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1:1–1:24, 2007.

17 Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666. ACM, 2001. doi:10.1145/380752.380867.

18 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.

19 Maxwell Herman Alexander Newman. On theories with a combinatorial definition of" equivalence". *Annals of mathematics*, pages 223–243, 1942.

20 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

21 Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14*, pages 80:1–80:6. ACM, 2014. doi:10.1145/2603088.2603121.