A parameterized view on the complexity of dependence and independence logic

Kontinen, Juha

2022-12

Kontinen , J , Meier , A & Mahmood , Y 2022 , ' A parameterized view on the complexity of dependence and independence logic ' , Journal of Logic and Computation , vol. 33 , no. 8 , þÿ e x a c 0 7 0 , pp. 1624 1644 . https://doi.org/10.1093/logcom/exac070

http://hdl.handle.net/10138/353242
https://doi.org/10.1093/logcom/exac070

A Parameterized View on the Complexity of Dependence and Independence Logic

Juha Kontinen ORCID
University of Helsinki, Department of Mathematics and Statistics, Helsinki, Finland

Arne Meier ORCID
Leibniz Universität Hannover, Institut für Theoretische Informatik, Hannover, Germany

Yasir Mahmood ORCID
Leibniz Universität Hannover, Institut für Theoretische Informatik, Hannover, Germany

Abstract

In this paper, we investigate the parameterized complexity of model checking for Dependence and Independence logic which are well studied logics in the area of Team Semantics. We start with a list of nine immediate parameterizations for this problem, namely: the number of disjunctions (i.e., splits)/(free) variables/universal quantifiers, formula-size, the tree-width of the Gaifman graph of the input structure, the size of the universe/team, and the arity of dependence atoms. We present a comprehensive picture of the parameterized complexity of model checking and obtain a division of the problem into tractable and various intractable degrees. Furthermore, we also consider the complexity of the most important variants (data and expression complexity) of the model checking problem by fixing parts of the input.

2012 ACM Subject Classification Theory of computation → Higher order logic; Theory of computation → Problems, reductions and completeness

Keywords and phrases Team Semantics, Dependence Logic, Independence Logic, Parameterized Complexity, Model Checking

Funding *Juha Kontinen*: Funded by grants 308712 and 338259 of the Academy of Finland
Arne Meier: Funded by the German Research Foundation (DFG), project ME4279/1-2
Yasir Mahmood: Funded by the German Research Foundation (DFG), project ME4279/1-2

1 Introduction

In this article, we explore the parameterized complexity of model checking for dependence $\mathcal{FO}(\mathsf{dep})$ and independence logic $\mathcal{FO}(\perp)$. We give a concise classification of this problem and its standard variants (expression and data complexity) with respect to several syntactic and structural parameters. Our results lay down a solid foundation for a systematic study of the parameterized complexity of team-based logics.

The introduction of dependence logic [?] in 2007 marks also the birth of the general semantic framework of team semantics that has enabled a systematic study of various notions of dependence and independence during the past decade. Team semantics differs from Tarski's semantics by interpreting formulas by sets of assignments instead of a single assignment as in first-order logic. Syntactically, dependence logic is an extension of first-order logic by new dependence atoms $\mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$ expressing that the values of variables $\boldsymbol{x}$ functionally determine values of the variables $\boldsymbol{y}$ (in the team under consideration). Similarly, independence logic is an extension of first-order logic by independence atoms $\boldsymbol{x}\perp_{\boldsymbol{z}}\boldsymbol{y}$ expressing that the values of variables $\boldsymbol{x}$ are independent of values of variables $\boldsymbol{y}$ for any given values of variables $\boldsymbol{z}$. Dependence and independence also manifest themselves in the context of database theory where one considers functional and multivalued dependencies [?].There are also other interesting team-based logics and atoms such as *inclusion* and *exclusion* atoms that are intimately connected to the corresponding inclusion and exclusion dependencies studied in database theory [?]. Furthermore, team semantics has been also extended, e.g., to propositional, modal and probabilistic variants (see [?, ?, ?] and the references therein).

For the applications, it is important to understand the complexity theoretic aspects of dependence logic and its variants. In fact, during the past few years, these aspects have been addressed in several studies. For example, on the level of sentences dependence logic and independence logic are equivalent to existential second-order logic while inclusion logic corresponds to positive greatest fixed point logic and thereby captures **P** over finite (ordered) structures [?]. Furthermore, there are (non-parameterized) studies that restrict the syntax and try to pin the intractability of a problem to a particular (set of) connective(s). For instance, Durand and Kontinen [?] characterize the data complexity of fragments of dependence logic with bounded arity of dependence atoms/number of universal quantifiers. For independence and inclusion logic, the similar characterization has been achieved by Kontinen et al. [?, ?]. Grädel [?] considered the combined and the expression complexity of the model checking problem of dependence and independence logic. These studies will be of great help in developing our parameterized approach.

A formalism to enhance the understanding of the inherent intractability of computational problems is brought by the framework of parameterized complexity [?]. Initiated by the founding fathers Downey and Fellows, in this area within computational complexity theory one strives for more structure within the darkness of intractability. Essentially, one tries to identify so-called parameters of a considered problem Π to find algorithms solving Π with runtimes of the form $f(k) \cdot |x|^{O(1)}$ for inputs x, corresponding parameter values k, and a computable function f. These kind of runtimes are called **FPT**-*runtimes* (from fixed-parameter tractable; short **FPT**) and tame the combinatoric explosion of the solution space to a function f in the parameter. As a very basic example in this vein, we can consider the propositional satisfiability problem SAT. An immediate parameter that pulls the problem into the class **FPT** is the number of variables, as one can solve SAT in time $2^k \cdot |\varphi|$ if k is the number of variables of a given propositional formula φ. Yet, this parameter is not very satisfactory as it neither is seen fixed nor slowly growing in its practical instances. However, there are several interesting other parameters under which SAT becomes fixed-parameter tractable, e.g., the so-called treewidth of the underlying graph representations of the considered formula [?]. This term was coined by Robertson and Seymour in 1984 [?] and established a profound position (currently DBLP lists 812 papers with treewidth in its title) also in the area of parameterized complexity in the last years [?, ?].

Coming back to fpt-runtimes, a runtime of a very different quality (yet still polynomial for fixed parameters) than **FPT** is summarized by the complexity class **XP**: $|x|^{f(k)}$ for inputs x, corresponding parameter values k, and a computable function f. Furthermore, analogously as **XP** but on nondeterministic machines, the class **XNP** will be of interest in this paper. Further up in the hierarchy, classes of the form $\mathbf{para}\mathcal{C}$ for a classical complexity class $\mathcal{C} \in \{\mathbf{NP}, \mathbf{PSPACE}, \mathbf{NEXP}\}$ play a role in this paper. Such classes intuitively capture all problems that are in the complexity class $\mathcal{C}$ after fpt-time preprocessing. In Fig. 1 an overview of these classes and their relations are depicted (for further details see, e.g., the work of Elberfeld et al. [?]).

Recently, the propositional variant of dependence logic ($\mathcal{PDL}$) has been investigated regarding its parameterized complexity [?, ?]. Moreover, propositional independence and inclusion logic have also been studied from the perspective of parameterized complexity [?]. In this paper, we further pursue the parameterized journey through the world of team logics and will visit the problems of first-order dependence $\mathcal{FO}(\mathsf{dep})$ and independence logic $\mathcal{FO}(\perp)$. As this paper is the first one that investigates these logics from the parameterized point of

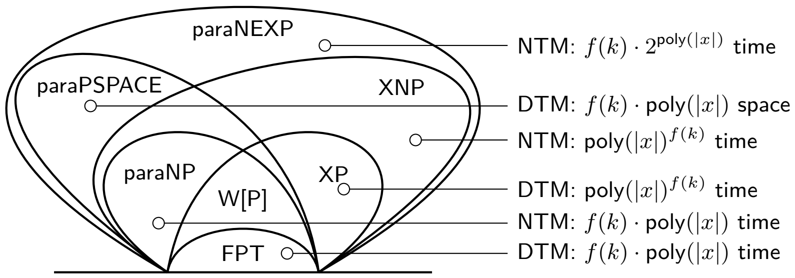

Figure 1 Landscape showing relations of relevant parameterized complexity classes with machine definitions.

Flight	Destination	Gate	Date	Time
FIN-70	HEL – FI	C1	04.10.2021	09:55
SAS-475	OSL – NO	A1	04.10.2021	12:25
SAS-476	HAJ – DE	A5	04.10.2021	12:25
FIN-80	HEL – FI	C1	04.10.2021	19:55
KLM-615	ATL – USA	A5	05.10.2021	11:55
THY-159	IST – TR	A1	05.10.2021	15:55
FIN-80	HEL – FI	C1	05.10.2021	19:55

Table 1 An example flight departure screen at an airport

view, we need to gather the existing literature and revisit many results particularly from this perspective. As a result, this paper can be seen as a systematic study with some of the result following in a straightforward manner from the known non-parameterized results and some shedding light also on the non-parameterized view of model checking.

We give an example below to illustrate how the concept of (in)dependence arises as a natural phenomenon in the physical world.

▶ **Example 1.** The database in Table 1 presents a screen at an airport for showing details about departing flights. Alternatively, it can be seen as a team T over attributes in the top row as variables. Clearly, $T \models \mathsf{dep}(\mathtt{Flight},\mathtt{Date},\mathtt{Time};\mathtt{Destination},\mathtt{Gate})$, as well as $T \models \mathsf{dep}(\mathtt{Gate},\mathtt{Date},\mathtt{Time};\mathtt{Destination},\ \mathtt{Flight})$.

Whereas, $T \not\models \mathsf{dep}(\mathtt{Destination},\mathtt{Gate};\mathtt{Time})$ as witnessed by the pair (FIN-70, HEL – FI, C1, 04.10.2021, 09:55) and (FIN-80, HEL – FI, C1, 04.10.2021, 19:55). Moreover, $T \models \mathtt{Gate}\perp_{\emptyset}\mathtt{Date}$, that is, the variable Gate is independent of Date when conditioned on empty set. Finally, $T \not\models \mathtt{Flight}\perp_{\mathtt{Date}}\mathtt{Time}$ as witnessed by the pair (FIN-70, HEL – FI, C1, 04.10.2021, 09:55) and (SAS-475, OSL – NO, A1, 04.10.2021, 12:25).

Contribution. Our classification is two-dimensional:

1. We consider the model checking problem of $\mathcal{FO}(\mathsf{dep})$ and $\mathcal{FO}(\perp)$ under various parameterizations: number of split-junctions in a formula #splits, the length of the formula $|\Phi|$, number of free variables #free-variables, the treewidth of the structure $\mathsf{tw}(\mathcal{A})$, the size of the structure $|\mathcal{A}|$, the size of the team $|T|$, the number of universal quantifiers in the formula $\#\forall$, the arity of the dependence atoms arity, as well as the total number of variables #variables.
2. We distinguish between expression complexity ec (the input structure is fixed), data complexity dc (the formula is fixed), and combined complexity cc.

The results are summarized in Table 2. For instance, the parameters #∀, arity, and #variables impact in lowering the complexity for ec (and not for cc or dc), while the parameter $|\mathcal{A}|$ impacts for dc but not for cc or ec.

Besides, we proved a general result on independence logic formulas that is independent of a parameterised analysis (Lemmas 11 and 14) and can be useful in other contexts.

Related work. The parameterized complexity analyses in the propositional setting [?, ?, ?] have considered the combined complexity of model checking and satisfiability as problems of interest. On the cc-level, the picture there is somewhat different, e.g., team size as a parameter for propositional dependence logic enabled a **FPT** algorithm while in our setting it has no effect on the complexity (**paraNEXP**). Grädel [?] studied the expression and the combined complexity for $\mathcal{FO}(\mathsf{dep})$ and $\mathcal{FO}(\perp)$ in the classical setting, whereas the data complexity was considered by Kontinen [?].

Prior work. This paper appeared in a preliminary version at the Logical Foundations of Computer Science (LFCS) 2022 Proceedings. In this version, we extend our complexity analysis to incorporate the strict and lax variant of independence logic. Lemmas 11 and 14 are new results.

Organization of the paper. In Section 2, we introduce the foundational concepts of dependence logic as well as parameterized complexity. In Section 3 our results are presented while Section 4 concludes the article.

2 Preliminaries

We require standard notions from classical complexity theory [?]. We encounter the classical complexity classes $\mathbf{P}, \mathbf{NP}, \mathbf{PSPACE}, \mathbf{NEXP}$ and their respective completeness notions, employing polynomial time many-one reductions ($\leq_m^{\mathbf{P}}$).

Parameterized Complexity Theory. A *parameterized problem* (PP) $P \subseteq \Sigma^* \times \mathbb{N}$ is a subset of the crossproduct of an alphabet and the natural numbers. For an *instance* $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the (value of the) *parameter*. A *parameterization* is a polynomial-time computable function that maps a value from $x \in \Sigma^*$ to its corresponding $k \in \mathbb{N}$. The problem P is said to be *fixed-parameter tractable* (or in the class **FPT**) if there exists a deterministic algorithm $\mathcal{A}$ and a computable function f such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$, algorithm $\mathcal{A}$ correctly decides the membership of $(x, k) \in P$ and runs in time $f(k) \cdot |x|^{O(1)}$. The problem P belongs to the class **XP** if $\mathcal{A}$ runs in time $|x|^{f(k)}$ on a deterministic machine, whereas **XNP** is the non-deterministic counterpart of **XP**. Abusing a little bit of notation, we write $\mathcal{C}$-machine for the type of machines that decide languages in the class $\mathcal{C}$, and we will say a function f is "$\mathcal{C}$-computable" if it can be computed by a machine on which the resource bounds of the class $\mathcal{C}$ are imposed.

Also, we work with classes that can be defined via a precomputation on the parameter.

▶ **Definition 2.** *Let $\mathcal{C}$ be any complexity class. Then* **para**$\mathcal{C}$ *is the class of all PPs $P \subseteq \Sigma^* \times \mathbb{N}$ such that there exists a computable function $\pi \colon \mathbb{N} \to \Delta^*$ and a language $L \in \mathcal{C}$ with $L \subseteq \Sigma^* \times \Delta^*$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ we have that $(x, k) \in P \Leftrightarrow (x, \pi(k)) \in L$.*

Notice that $\mathbf{paraP} = \mathbf{FPT}$. The complexity classes $\mathcal{C} \in \{\mathbf{NP}, \mathbf{PSPACE}, \mathbf{NEXP}\}$ are used in the **para**$\mathcal{C}$ context by us.

A problem P is in the complexity class $\mathbf{W[P]}$, if it can be decided by a NTM running in time $f(k) \cdot |x|^{O(1)}$ steps, with at most $g(k)$-many non-deterministic steps, where f, g are computable functions. Moreover, $\mathbf{W[P]}$ is contained in the intersection of **paraNP** and **XP** (for details see the textbook of Flum and Grohe [?]).

Let $c \in \mathbb{N}$ and $P \subseteq \Sigma^* \times \mathbb{N}$ be a PP, then the *c-slice of* P, written as P_c is defined as $P_c := \{ (x, k) \in \Sigma^* \times \mathbb{N} \mid k = c \}$. Notice that P_c is a classical problem then. Observe that, regarding our studied complexity classes, showing membership of a PP P in the complexity class $\mathbf{para}\mathcal{C}$, it suffices to show that for each slice $P_c \in \mathcal{C}$ is true.

▶ **Definition 3.** *Let $P \subseteq \Sigma^* \times \mathbb{N}, Q \subseteq \Gamma^*$ be two PPs. One says that P is* fpt-reducible *to Q, $P \leq^{\mathbf{FPT}} Q$, if there exists an $\mathbf{FPT}$-computable function $f\colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$ such that*

- *for all $(x, k) \in \Sigma^* \times \mathbb{N}$ we have that $(x, k) \in P \Leftrightarrow f(x, k) \in Q$,*
- *there exists a computable function $g\colon \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ and $f(x, k) = (x', k')$ we have that $k' \leq g(k)$.*

Finally, in order to show that a problem P is $\mathbf{para}\mathcal{C}$-hard (for some complexity class $\mathcal{C}$) it is enough to prove that for some $c \in \mathbb{N}$, the slice P_c is $\mathcal{C}$-hard in the classical setting.

Dependence and Independence Logic. We assume basic familiarity with predicate logic [?]. We consider first-order vocabularies τ that are sets of *function* symbols and *relation* symbols with an equality symbol $=$. Let VAR be a countably infinite set of *first-order variables*. Terms over τ are defined in the usual way, and the set of well-formed formulas of first order logic ($\mathcal{FO}$) is defined by the following BNF:

$$\psi ::= t_1 = t_2 \mid R(t_1, \dots, t_k) \mid \neg R(t_1, \dots, t_k) \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x \psi \mid \forall x \psi,$$

where t_i are terms $1 \leq i \leq k$, R is a k-ary relation symbol from σ, $k \in \mathbb{N}$, and $x \in \mathrm{VAR}$. If ψ is a formula, then we use $\mathrm{VAR}(\psi)$ for its set of variables, and $\mathrm{Fr}(\psi)$ for its set of free variables. We evaluate $\mathcal{FO}$-formulas in τ-structures, which are pairs of the form $\mathcal{A} = (A, \tau^{\mathcal{A}})$, where A is the *domain* of $\mathcal{A}$ (when clear from the context, we write A instead of $\mathrm{dom}(\mathcal{A})$), and $\tau^{\mathcal{A}}$ interprets the function and relational symbols in the usual way (e.g., $t^{\mathcal{A}}\langle s \rangle = s(x)$ if $t = x \in \mathrm{VAR}$). If $\boldsymbol{t} = (t_1, \dots, t_n)$ is a tuple of terms for $n \in \mathbb{N}$, then we write $\boldsymbol{t}^{\mathcal{A}}\langle s \rangle$ for $(t_1^{\mathcal{A}}\langle s \rangle, \dots, t_n^{\mathcal{A}}\langle s \rangle)$.

Dependence logic ($\mathcal{FO}(\mathsf{dep})$) extends $\mathcal{FO}$ by dependence atoms of the form $\mathsf{dep}(\boldsymbol{t}; \boldsymbol{u})$ where $\boldsymbol{t}$ and $\boldsymbol{u}$ are tuples of terms. Independence logic ($\mathcal{FO}(\perp)$) in obtained by adding to $\mathcal{FO}$ the independence atoms of the form $\boldsymbol{t} \perp_{\boldsymbol{v}} \boldsymbol{u}$ for tuples $\boldsymbol{t}, \boldsymbol{u}$ and $\boldsymbol{v}$ of terms. We call expressions of the kind $t_1 = t_2, R(\boldsymbol{t}), \mathsf{dep}(\boldsymbol{t}; \boldsymbol{u})$, and $\boldsymbol{t} \perp_{\boldsymbol{v}} \boldsymbol{u}$ *atomic formulas*.

The semantics is defined through the concept of a team. Let $\mathcal{A}$ be a structure and $X \subseteq \mathrm{VAR}$, then an *assignment* s is a mapping $s\colon X \to A$.

▶ **Definition 4.** *Let $X \subseteq \mathrm{VAR}$. A* team *T in $\mathcal{A}$* with domain *X is a set of assignments $s\colon X \to A$.*

For a team T with domain $X \supseteq Y$ define its *restriction* to Y as $T \upharpoonright Y := \{ s \upharpoonright Y \mid s \in T \}$. If $s\colon X \to A$ is an assignment and $x \in \mathrm{VAR}$ is a variable, then $s^x_a\colon X \cup \{x\} \to A$ is the assignment that maps x to a and $y \in X \setminus \{x\}$ to $s(y)$. Let T be a team in $\mathcal{A}$ with domain X. Then any function $f\colon T \to \mathcal{P}(A) \setminus \{\emptyset\}$ can be used as a *supplementing function* of T to extend or modify T to the *supplemented team* $T^x_f := \{ s^x_a \mid s \in T, a \in f(s) \}$. For the case $f(s) = A$ is the constant function we simply write $T^x_{\mathcal{A}}$ for T^x_f. The semantics of formulas is defined as follows.

▶ **Definition 5.** *Let τ be a vocabulary, $\mathcal{A}$ be a τ-structure and T be a team over $\mathcal{A}$ with domain $X \subseteq \mathrm{VAR}$. Then,*

$$
\begin{array}{lll}
(\mathcal{A},T) \models t_1 = t_2 & \textit{iff} & \forall s \in T : t_1^{\mathcal{A}}\langle s\rangle = t_2^{\mathcal{A}}\langle s\rangle \\
(\mathcal{A},T) \models R(t_1,\dots,t_n) & \textit{iff} & \forall s \in T : (t_1^{\mathcal{A}}\langle s\rangle,\dots,t_n^{\mathcal{A}}\langle s\rangle) \in R^{\mathcal{A}} \\
(\mathcal{A},T) \models \neg R(t_1,\dots,t_n) & \textit{iff} & \forall s \in T : (t_1^{\mathcal{A}}\langle s\rangle,\dots,t_n^{\mathcal{A}}\langle s\rangle) \notin R^{\mathcal{A}} \\
(\mathcal{A},T) \models \mathsf{dep}(\boldsymbol{t};\boldsymbol{u}) & \textit{iff} & \forall s_1,s_2 \in T : \boldsymbol{t}^{\mathcal{A}}\langle s_1\rangle = \boldsymbol{t}^{\mathcal{A}}\langle s_2\rangle \implies \boldsymbol{u}^{\mathcal{A}}\langle s_1\rangle = \boldsymbol{u}^{\mathcal{A}}\langle s_2\rangle \\
(\mathcal{A},T) \models \boldsymbol{t}\perp_{\boldsymbol{v}}\boldsymbol{u} & \textit{iff} & \forall s_1,s_2 \in T : \boldsymbol{v}^{\mathcal{A}}\langle s_1\rangle = \boldsymbol{v}^{\mathcal{A}}\langle s_2\rangle \textit{ then } \exists s_3 \in T : \\
 & & \quad \boldsymbol{vt}^{\mathcal{A}}\langle s_3\rangle = \boldsymbol{vt}^{\mathcal{A}}\langle s_1\rangle \textit{ and } \boldsymbol{u}^{\mathcal{A}}\langle s_3\rangle = \boldsymbol{u}^{\mathcal{A}}\langle s_2\rangle \\
(\mathcal{A},T) \models \phi_0 \wedge \phi_1 & \textit{iff} & (\mathcal{A},T) \models \phi_0 \quad \textit{and} \quad (\mathcal{A},T) \models \phi_1 \\
(\mathcal{A},T) \models \phi_0 \vee \phi_1 & \textit{iff} & \exists T_0 \exists T_1 : T_0 \cup T_1 = T \quad \textit{and} \quad (\mathcal{A},T_i) \models \phi_i \textit{ for } i = 0,1 \\
(\mathcal{A},T) \models \exists x\phi & \textit{iff} & (\mathcal{A},T_f^x) \models \phi \textit{ for some } f\colon T \to \mathcal{P}(A) \setminus \{\emptyset\} \\
(\mathcal{A},T) \models \forall x\phi & \textit{iff} & (\mathcal{A},T_{\mathcal{A}}^x) \models \phi
\end{array}
$$

Notice that we only consider formulas in negation normal form (NNF). In the team semantics setting, disjunction and existential quantifier are given two different meanings. The above defined semantics is the so-called *lax*-semantics, whereas an alternative is the *strict*-semantics. In strict-semantics, the split of teams have to be disjoint and the supplementing function is replaced by a function $f\colon T \to A$. That is, the function f assigns a single element $a \in A$ to each $s \in T$. For dependence logic, the two semantics coincide due to the downwards closure property. That is, for any $\mathcal{FO}(\mathsf{dep})$-formula ϕ, if $(\mathcal{A},T) \models \phi$ then $(\mathcal{A},P) \models \phi$ for every $P \subseteq T$. For this reason we only consider lax semantics for $\mathcal{FO}(\mathsf{dep})$. Further note that $(\mathcal{A},T) \models \phi$ for all ϕ when $T = \emptyset$ (this is also called the *empty team property*). Finally, $\mathcal{FO}(\mathsf{dep})$-formulas are *local*, that is, for a team T in $\mathcal{A}$ over domain X and a $\mathcal{FO}(\mathsf{dep})$-formula ϕ, we have that $(\mathcal{A},T) \models \phi$ if and only if $(\mathcal{A},T \upharpoonright \mathrm{Fr}(\phi)) \models \phi$. $\mathcal{FO}(\perp)$-formulas are also local under lax-semantics but not under strict-semantics [**?**, Prop. 4.7]. Notice that strict-semantics is relatively stricter (as the name suggest) than the lax-semantics [**?**]. That is, for every $\mathcal{FO}(\perp)$-formula ϕ, if $(\mathcal{A},T) \models_s \phi$ then $(\mathcal{A},T) \models_\ell \phi$, where the subscript s and ℓ indicates the choice of the semantics. As a consequence, our hardness results for lax-semantics also apply to the case of strict-semantics. However, for membership we need to consider them separately for each case.

▶ **Definition 6** (Gaifman graph)**.** *Given a vocabulary τ and a τ-structure $\mathcal{A}$, the Gaifman graph $G_{\mathcal{A}} = (A,E)$ of $\mathcal{A}$ is defined as*

$$E := \big\{\, \{u,v\} \;\big|\; \textit{if there is an } R^n \in \tau \textit{ and } \boldsymbol{a} \in A^n \textit{ with } R^{\mathcal{A}}(\boldsymbol{a}) \textit{ and } u,v \in \boldsymbol{a} \,\big\}.$$

That is, there is a relation $R \in \tau$ of arity n such that u and v appear together in $R^{\mathcal{A}}$.

Intuitively, the Gaifman graph of a structure $\mathcal{A}$ is an undirected graph with the universe of $\mathcal{A}$ as vertices and connects two vertices when they share a tuple in a relation (see also Fig. 2).

▶ **Definition 7** (Treewidth)**.** *The* tree decomposition *of a given graph $G = (V,E)$ is a tree $T = (B,E_T)$, where the vertex set $B \subseteq \mathcal{P}(V)$ is the collection of* bags *and E_T is the edge relation such that the following is true.*

- $\bigcup_{b\in B} b = V$,
- *for every $\{u,v\} \in E$ there is a bag $b \in B$ with $u,v \in b$, and*
- *for all $v \in V$ the restriction of T to v (the subset with all bags containing v) is connected.*

Flight	Gate	Time
FIN-**7**0	C1	**09**:55
SAS-47**5**	C3	**12**:25
SAS-47**6**	C2	**12**:25
FIN-**8**0	C1	**19**:55

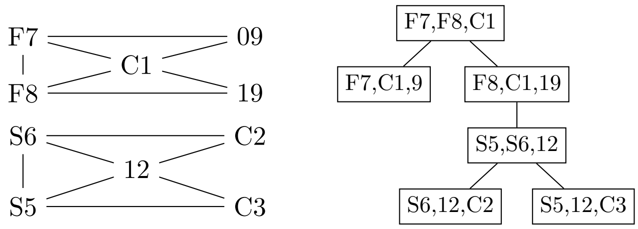

Figure 2 An $\mathcal{FO}$-structure $\mathcal{A} = (A, S^{\mathcal{A}}, R^{\mathcal{A}})$ (Left) with the Gaifman graph $G_{\mathcal{A}}$ (Middle) and a possible treedecomposition of $G_{\mathcal{A}}$ (Right) of Example 8. For brevity, universe elements are written in short forms.

The width *of a given tree decomposition $T = (B, E_T)$ is the size of the largest bag minus one: $\max_{b \in B} |b| - 1$. The* treewidth *of a given graph G is the minimum over all widths of tree decompositions of G.*

Observe that if G is a tree then the treewidth of G is one. Intuitively, one can say that treewidth accordingly is a measure of tree-likeness of a given graph.

▶ **Example 8.** Consider the database form our previous example. Recall that the universe A consists of entries in each row. Let $\tau = \{\mathrm{S}^2, \mathrm{R}^3\}$ include a binary relation S ($\mathrm{S}(x, y)$: flights x and y are owed by the same company) and a ternary relation R ($\mathrm{R}(x, y, z)$: the gate x is reserved by the flight y at time z). For simplicity, we only consider first four rows with the corresponding three columns from Table 4, see Figure 2 for an explanation. Since the largest bag size in our decomposition is 3, the treewidth of this decomposition is 2. Furthermore, the presence of cycles of length 3 suggests that there is no better decomposition. As a consequence the given structure has treewidth 2.

The decision problem to determine whether the treewidth of a given graph $\mathcal{G} = (V, E)$ is at most k, is **NP**-complete [**?**]. See Bodlaender's Guide [**?**] for an overview of algorithms that compute tree decompositions. When considering the parameter treewidth, one usually assumes it as a given value and does not need to compute it.

In the following problem definitions let $\mathcal{C} \in \{\mathcal{FO}(\mathsf{dep}), \mathcal{FO}(\perp)\}$. We consider only the model checking problem (MC) and two variants in this paper. First, let us define the most general version.

Problem:	$\mathsf{cc}(\mathcal{C})$ (combined complexity of model checking)
Input:	a structure $\mathcal{A}$, team T and a $\mathcal{C}$-formula Φ.
Question:	$(\mathcal{A}, T) \models \Phi$?

We further consider the following two variants of the model checking problem.

Problem:	$\mathsf{dc}(\mathcal{C})$ (data complexity of model checking, $\mathcal{C}$-formula Φ is fixed)
Input:	a structure $\mathcal{A}$, team T.
Question:	$(\mathcal{A}, T) \models \Phi$?

Problem:	$\mathsf{ec}(\mathcal{C})$ (expression complexity of model checking, $\mathcal{A}, T$ are fixed)
Input:	a $\mathcal{C}$-formula Φ.
Question:	$(\mathcal{A}, T) \models \Phi$?

Parameter	cc	dc	ec
#splits	**paraPSPACE**-hard[L25]	**paraNP**[L20]	**paraPSPACE**-hard[L25]
$\lvert\Phi\rvert$	**paraNP**[L26]	**paraNP**[R21]	**FPT**[27]
#free-variables	**paraNEXP**[L24]	**paraNP**[L20]	**paraNEXP**[L24]
tw($\mathcal{A}$)	**paraNEXP**[L24]	**paraNP**[P19]	**paraNEXP**[L24]
$\lvert\mathcal{A}\rvert$	**paraNEXP**[L24]	**FPT**[L22]	**paraNEXP**[L24]
$\lvert T\rvert$	**paraNEXP**[L24]	**paraNP**[L23]	**paraNEXP**[L24]
#$\forall$	**paraNP**-hard[L30]	**paraNP**[L20]	**paraNP**[L28]
arity	**paraPSPACE**-hard[L33]	**paraNP**[L20]	**paraPSPACE**[L31]
#variables	**paraNP**[L35]	**paraNP**[L20]	**FPT**[L36]

Table 2 Complexity classification overview for both logics. The numbers in the exponent point to the corresponding result (Lx means Lemma x, Px means Proposition x, Rx means Remark x). Fig. 3 on page 18 is a graphical presentation of this table with a different angle.

List of Parameterizations. Now let us turn to the parameters that are under investigation in this paper. We study the model checking problem of $\mathcal{C}$ under nine various parameters that naturally occur in an MC-instance. Let $\langle \mathcal{A}, T, \Phi\rangle$ be an instance of MC, where Φ is a $\mathcal{C}$-formula, $\mathcal{A}$ is a structure and T is a team over $\mathcal{A}$. The parameter #splits denotes the number of occurrences of the split operator ($\vee$), #$\forall$ is the number of universal quantifiers in Φ. Moreover, #variables (resp., #free-variables) denotes the total number of (free) variables in Φ. The parameter $|\Phi|$ is the size of the input formula Φ, and similarly the two other size parameters are $|\mathcal{A}|$ and $|T|$. The treewidth of the structure $\mathcal{A}$ (see Def. 7) is defined as the treewidth of $G_{\mathcal{A}}$ and denoted by tw($\mathcal{A}$). Note that for formulas using the dependence atom dep($\boldsymbol{x}; \boldsymbol{y}$), one can translate to a formula using only dependence atoms where $|\boldsymbol{y}| = 1$ (via conjunctions). That is why the arity of a dependence atom dep($\boldsymbol{x}; \boldsymbol{y}$) is defined as $|\boldsymbol{x}|$. The arity of an independence atom $\boldsymbol{x} \perp_{\boldsymbol{z}} \boldsymbol{y}$ is defined as $|\boldsymbol{x} \cup \boldsymbol{y} \cup \boldsymbol{z}|$. Finally, arity is the maximum arity of any dependence (independence) atom in Φ. Let k be any parameterization and $P \in \{$dc, ec, cc$\}$, then by k-P we denote the problem P when parameterized by k. If more than one parameterization is considered, then we use '+' as a separator and write these parameters in brackets, e.g., ($|\Phi|$+#free-variables)-dc as the problem dc with parameterization $|\Phi|$ + #free-variables. Finally, notice that since the formula Φ is fixed for dc this implies that $|\Phi|$-dc is nothing but dc. That is, bounding the parameter does not make sense for dc as the problem dc remains **NP**-complete.

3 Complexity results

We begin by proving several relationships between various parameterizations. These results are true for both $\mathcal{FO}$(dep) and $\mathcal{FO}(\perp)$.

▶ **Lemma 9.** *Let Φ be a $\mathcal{C}$-formula, $\mathcal{A}$ be a structure and T a team over $\mathcal{A}$. Then the following relations among parameters hold.*

1. $|\Phi| \geq k$ *for any* $k \in \{$ #splits, #$\forall$, arity, #free-variables, #variables $\}$,
2. $|\mathcal{A}| \geq$ tw($\mathcal{A}$). *Moreover, for* dc, $|\mathcal{A}|^{O(1)} \geq |T|$,
3. *For* ec, #free-variables *is constant.*

Proof. **1.** Clearly, the size of the formula limits all parts of it including the parameters mentioned in the list.

	x	y	z	t	u
s_0	0	0	0	1	1
s_1	0	1	1	0	1
s_2	1	1	0	1	0
s_3	1	0	0	1	1
s_4	0	1	0	1	0
s_5	0	1	0	0	1
s_6	0	0	1	1	1

Table 3 An example team that satisfies the formula in Example 10 in lax-semantics, but not in strict-semantics.

2. Notice that for data complexity, the formula Φ and consequently the number of free variables in Φ is fixed. Moreover, due to locality principle it holds that $T \subseteq A^r$, where r is the number of free variables in Φ. That is, the team T can be considered only over the free variables of Φ. This implies that teamsize is polynomially bounded by the universe size, as $|T| \leq |\mathcal{A}|^r$. Notice that $\mathcal{FO}(\perp)$ with strict-semantics does not satisfy locality. Consequently, the aforementioned proof works for $\mathcal{FO}(\mathsf{dep})$-formulas, but only for lax semantics in the context of $\mathcal{FO}(\perp)$-formulas.
Finally, the result for $\mathsf{tw}(\mathcal{A})$ follows due to Definition 7. This is due to the reason that in the worst case all universe elements belong to one bag in the decomposition and $\mathsf{tw}(\mathcal{A}) = |\mathcal{A}| - 1$.

3. Notice that the team T is fixed in ec. This implies that the domain of T (which contains the set of free variables in the formula Φ) is also fixed and as a result, #free-variables is constant. ◀

As discussed before, $\mathcal{FO}(\mathsf{dep})$-formulas are local in the sense that: given a team T and a formula Φ then $T \models \Phi$ iff $T \upharpoonright_{\mathsf{VAR}(\Phi)} \models \Phi$. Moreover, $\mathcal{FO}(\perp)$-formulas are also local but only under lax-semantics. The locality fails for strict semantics due to the reason that there might exist two assignments $s, t \in T$ such that $s \neq t$ and $s(v) = t(v)$ for each $v \in \mathsf{VAR}(\Phi)$. If we restrict T to $\mathsf{VAR}(\Phi)$ then s and t collapse into just one assignment restricting the ways in which a team can be split into two disjoint parts.

▶ **Example 10.** Consider the formula $\phi = (x \perp y \wedge z \neq t) \vee (y \perp z \wedge x \neq u)$ and the team T as depicted in Table 3. Clearly, $\{s_0, s_1, s_2, s_3, s_4\} \models x \perp y \wedge z \neq t$ and $\{s_0, s_1, s_2, s_5, s_6\} \models y \perp z \wedge x \neq u$, thereby $T \models_\ell \phi$. Whereas, s_3, s_4 must be in the left split and s_5, s_6 must be in the right split. Moreover, we can add s_2 to the left split and s_1 to the right. Now, s_1 must be in both splits in order for the independence atoms to be true but this is not allowed in strict semantics.

As Example 10 depicts, the question whether $T \models \Phi$ cannot be reduced to the question whether $T \upharpoonright_{\mathsf{VAR}(\Phi)}$ in the strict-semantics. As a consequence, for $\mathcal{FO}(\perp)$-formulas under strict semantics when T is part of the input the size of T cannot be directly bounded by other parameters. However, when $|\mathcal{A}| = k$ is the parameter and $|\Phi|$ is fixed (for data complexity), the following lemma applies.

▶ **Lemma 11.** *Let Φ be an $\mathcal{FO}(\perp)$-formula with $\mathsf{VAR}(\Phi) = V$, $\mathcal{A}$ be a structure and T be a team in $\mathcal{A}$ over variables X. Then it is possible to construct in time polynomial in the size of Φ, $\mathcal{A}$ and T a formula Φ', a structure $\mathcal{A}'$ and a team T' over $V \cup \{z\}$, where $z \notin V$, such that $(\mathcal{A}, T) \models_s \Phi$ iff $(\mathcal{A}', T') \models_s \Phi'$.*

Proof. The idea is to simulate the multiplicity of assignments in $s \in T \upharpoonright_V$ by an additional variable z. Let ℓ be the the largest multiplicity of any assignment $s \in T \upharpoonright_V$. That is, let $\ell_s = \#\{t \mid t \in T \text{ and } t \upharpoonright_V = s\}$ and $\ell = \max\{\ell_s \mid s \in T\}$. In order to count up assignments in T' we add ℓ additional elements to $\mathcal{A}'$. This can be problematic for quantifiers in Φ as those now range over elements in $\mathcal{A}'$ rather than elements of $\mathcal{A}$. We avoid this by adding a unary relation symbol P such that $P^{\mathcal{A}'}$ is true only for these new elements. Let $\{a_1, \dots, a_\ell\}$ be a collection of fresh elements and consider the structure $\mathcal{A}' = (A \cup \{a_1, \dots, a_\ell\}, P^{\mathcal{A}'})$ where P is a unary relation as described above. First we construct the team T' from T by considering each collection $s^i_1 \dots, s^i_{r_i} \in T$ of assignments that agree over V and extending it in such a way that $s^i_j(z) = a_j$. Clearly, $j \leq \ell$ by construction. Notice that $\ell \leq |T|$ and therefore, the construction can be achieved in polynomial time. Moreover, $|T| = |T'|$. Now we construct the formula Φ' from Φ. It suffices to replace only the quantifiers. That is, $\forall x\psi$ is replaced by $\forall x(P(x) \vee (\neg P(x) \wedge \psi'))$ and $\exists x\psi$ is replaced by $\exists x(\neg P(x) \wedge \psi')$. The intuition for universal quantifier is that once each assignment in T' have been supplemented by $\mathcal{A}'$, we ignore those assignments which map x to $\{a_1, \dots, a_\ell\}$ because the quantified variable x in Φ ranges over elements of $\mathcal{A}$ alone. Similarly, for the case of existential quantifiers we assure that the supplementing function takes values only over $\mathcal{A}$ and not over $\mathcal{A}'$.

Now we prove the correctness by an induction on Φ for all T and T' as above. The case when Φ is a literal is easy because atomic formulas and their negations satisfy locality in both semantics. When $\Phi = \psi_0 \wedge \psi_1$, then the claim follows due to the induction hypothesis. Now we prove the claim for $\Phi = \psi_0 \vee \psi_1$. Clearly, $(\mathcal{A}, T) \models_s \Phi$ iff $\exists T_0 T_1$ such that $T_0 \uplus T_1 = T$ (that is, $T_0 \cup T_1 = T, T_0 \cap T_1 = \emptyset$) and $(\mathcal{A}, T_i) \models_s \psi_i$ for $i = 0, 1$. But we can use subteams T_i to construct subteams T'_i of T' such that $T'_0 \cup T'_1 = T', T'_0 \cap T'_1 = \emptyset$ and $(\mathcal{A}', T'_i) \models_s \psi'_i$ by induction hypothesis. This is due to the reason that $|T| = |T'|$ and there is a 1-1-correspondence ($g\colon T \to T'$) between T and T'. Consequently, the claim follows. Now, let $\Phi = \exists x\phi$. Then there is a function $f\colon T \to A$ such that $(\mathcal{A}, T^x_f) \models_s \phi$. But then consider the function $f'\colon T' \to A'$ such that for each $s \in T'$, $f'(s) = f(g^{-1}(s))$. Clearly, $f'(s) \in A$ and $(\mathcal{A}', T'^x_{f'}) \models_s \exists x(\neg P(x) \wedge \phi')$ and consequently $(\mathcal{A}', T') \models_s \Phi'$. The reverse direction follows a similar argument since $(\mathcal{A}', T') \models_s \exists x(\neg P(x) \wedge \phi')$ implies that the supplementing function $f'\colon T' \to A'$ is allowed to take only elements in A because of the subformula $\neg P(x)$. This together with the bijection g gives a supplementing function f such that $(\mathcal{A}, T^x_f) \models_s \phi$. Finally, the case when $\Phi = \forall x\phi$ is similar. ◀

We extract the following definition from the proof of Lemma 11.

▶ **Definition 12.** *Let T be a team, V be a set of variables, and $s \in T$ be an assignment. Then define $\ell_s = |\{\, t \mid t \in T$ and $t \upharpoonright_V = s \,\}|$ as the* multiplicity *of s.*

It is important to notice that the number ℓ of repeating assignments is neither bounded by $\mathcal{A}$ nor by $|\Phi|$ but by the multiplicity of assignments in T. It turns out that we can not directly bound the teamsize by the structure size and the size of the formula alone. However, with the following observation we can still achieve an upper bound. The idea is to determine the maximum multiplicity of each assignment required to evaluate a subformula in Φ, where we count the multiplicity with respect to $\mathrm{Fr}(\Phi)$ rather than only with respect to the variables in subformulas $\phi \in \mathsf{SF}(\Phi)$. That is, we do not restrict the multiplicity of assignments with respect to $\mathsf{VAR}(\phi)$ because $\mathrm{Fr}(\Phi)$ suffices for our purpose. Intuitively, for an atomic $\phi \in \mathsf{SF}(\Phi)$ it is enough to consider each assignment over $\mathrm{Fr}(\Phi)$ only once. The case of conjunction is simple because the team is the same for both conjuncts and therefore it is enough to take the maximum multiplicity for assignments in any conjunct. The interesting cases are split junction and the existential quantifier. If a subformula ϕ_i requires the multiplicity of an assignment s to be r_i for $i = 0, 1$, then clearly $\phi_0 \vee \phi_1$ requires the multiplicity of s to be $r_0 + r_1$. This is due to the reason that the considered subteam P for $\phi_0 \vee \phi_1$ can then split (according to the strict semantics) into subteams P_1 and P_2 with their corresponding multiplicities. Moreover, for $\exists x\phi$ the analysis takes into consideration the worst case scenario. That is, where the supplementing function for a strict existential quantifier takes only one value for x. In the worst case, there may be so many assignments that x can take each element a of the universe. As a consequence, the multiplicity of assignments increases by $|\mathcal{A}|$ (in principle, this can increase to $\min\{\ell, |\mathcal{A}|\}$ but we want to relate it with $|\mathcal{A}|$). Finally, the case of $\forall x\phi$ is simple because the supplementing function will map x to each element of the universe under each assignment.

▶ **Definition 13.** *Let Φ be an $\mathcal{FO}(\bot)$-formula with $\mathsf{VAR}(\Phi) = V$, $\mathcal{A}$ be a structure and T be a team with domain $X \supseteq V$. Define the function $f_\#\colon \mathsf{SF}(\Phi) \to \mathbb{N}$ such that*
1. *$f_\#(\phi) = 1$ for each atomic ϕ,*
2. *$f_\#(\phi \wedge \psi) = \max\{f_\#(\phi), f_\#(\psi)\}$,*
3. *$f_\#(\phi \vee \psi) = f_\#(\phi) + f_\#(\psi)$,*
4. *$f_\#(\exists x\phi) = f_\#(\phi) + |\mathcal{A}|$,*
5. *$f_\#(\forall \phi) = f_\#(\phi)$.*

The value $f_\#(\phi)$ assigns the maximum multiplicity of any assignment in a team T that might be required to evaluate $T \models \phi$.

▶ **Example 14.** Consider a team T and the formula $\Phi := \exists x\forall y[\phi_2(x, y, z) \wedge (\phi_0(x, y, z) \vee \phi_1(x, y))]$ where ϕ_i is atomic for each $i \leq 2$. This implies each ϕ_1 is local and therefore $f_\#(\phi_i) = 1$. Moreover, $f_\#(\phi_0 \vee \phi_1) = 2$, $f_\#(\phi_1 \wedge (\phi_0 \vee \phi_1)) = 2$ and $f_\#(\forall y\phi_1 \wedge (\phi_0 \vee \phi_1)) = 2$. Finally, $f_\#(\Phi) = 2 + s$ where $s = \min\{\ell, |\mathcal{A}|\}$ and ℓ is the maximum multiplicity of any assignment $s \in T \upharpoonright_V$.

The following lemma is essential in bounding the teamsize for data complexity of $\mathcal{FO}(\bot)$ under strict-semantics in terms of $|\mathcal{A}|$.

▶ **Lemma 15.** *Let Φ be an $\mathcal{FO}(\bot)$-formula with $\mathsf{VAR}(\Phi) = V$, $\#\mathsf{splits}(\Phi) = r$, $\#\exists(\Phi) = q$, $\mathcal{A}$ be a structure and T be a team in $\mathcal{A}$ over X. Then the following two claims are true:*
1. *$f_\#(\Phi) \leq (r + 1) + q \cdot |\mathcal{A}|$.*
2. *Let $T' \subseteq T$ be a team such that in T' each assignment $s \in T \upharpoonright_V$ has a multiplicity of at most $f_\#(\Phi)$. Then, we have that $(\mathcal{A}, T) \models_s \Phi$ iff $(\mathcal{A}, T') \models_s \Phi$. Furthermore, such a team T' can be computed in polynomial time in $|T|$.*

Proof. The claim that $f_\#(\Phi) \leq (r + 1) + q \cdot |\mathcal{A}|$ is easy to observe since $f_\#(\phi)$ only changes when $\phi = \psi_0 \vee \psi_1$ or $\phi = \exists x\psi$. In the first case, we take the sum for each split and in the second case, we add a factor of $|\mathcal{A}|$ for each existential quantifier.

To prove the second claim, notice first that if each assignment $s \in T \upharpoonright_V$ has already a multiplicity of at most $f_\#(\Phi)$ then there is nothing to prove and we take $T' = T$. Now, we show using induction on Φ that for all T', T satisfying for each assignment $s \in T \upharpoonright_V$ that either s has the same multiplicity, or a multiplicity of at least $f_\#(\Phi)$ in both of them, this implies that

$$(\mathcal{A}, T) \models \Phi \Leftrightarrow (\mathcal{A}, T') \models \Phi.$$

If Φ is an atomic or negated atomic formula then the claim follows from the fact that $T \upharpoonright_V = T' \upharpoonright_V$. Assume then that $\Phi = \psi_1 \vee \psi_2$ and T and T' satisfy the assumption on the number of extensions of assignments for $f_\#(\Phi) = n$. Then, $(\mathcal{A}, T) \models_s \Phi$ iff $\exists T_0, \exists T_1$, such that $T_0 \uplus T_1 = T$ and $(\mathcal{A}, T_i) \models_s \psi_i$ for $i = 0, 1$. It is now straightforward to check that we can define a partition of T' into T'_1 and T'_2 such that for all $s \in T_i \upharpoonright_V$ either s has the same multiplicities in T_i and T'_i, or multiplicities of at least $f_\#(\psi_i)$ in both of them. By the induction assumption it follows that $(\mathcal{A}, T'_i) \models \psi_i$. The converse implication is proved symmetrically. The other connectives can be treated in the same way. ◀

Lemma 14 results in bounding the size of an input team T by a constant factor of a polynomial in $|\mathcal{A}|$. The following corollary essentially provides the counterpart of second item in Lemma 9 for strict semantics of $\mathcal{FO}(\perp)$.

▶ **Corollary 16.** *Let Φ be an $\mathcal{FO}(\perp)$-formula with $\mathsf{VAR}(\Phi) = V$, $\#\mathsf{splits}(\Phi) = r$, $\#\exists(\Phi) = q$, $\mathcal{A}$ be a structure and T be a team in $\mathcal{A}$ over variables X. Then there is a team T' with $|T'| \leq (r + 1 + q \cdot |\mathcal{A}|) \cdot |T \upharpoonright_V|$ such that $T \models_s \Phi$ iff $T' \models_s \Phi$.*

Proof. For each assignment $s \in T \upharpoonright_V$, it is enough to consider at most $f_\#(\Phi)$ extensions of s. This yields the desired bound on the size of T'. ◀

▶ Remark 17. If the number of free variables ($\#\mathsf{free\text{-}variables}$) in a formula Φ is bounded then the total number of variables ($\#\mathsf{variables}$) is not necessarily bounded, on the other hand, bounding $\#\mathsf{variables}$ also bounds $\#\mathsf{free\text{-}variables}$.

Now we explore the relationship between $\mathcal{FO}(\mathsf{dep})$ and $\mathcal{FO}(\perp)$ which is essential in proving hardness results for $\mathcal{FO}(\perp)$.

▶ **Observation 18.** *The equivalence $\mathsf{dep}(\boldsymbol{x}; \boldsymbol{y}) \equiv \boldsymbol{y} \perp_{\boldsymbol{x}} \boldsymbol{y}$ between dependence and independence atoms implies $\mathcal{FO}(\mathsf{dep})$ can be viewed as a sublogic of $\mathcal{FO}(\perp)$. As a consequence, (in the classical setting) the hardness results for $\mathcal{FO}(\mathsf{dep})$ immediately translate to those for $\mathcal{FO}(\perp)$.*

Nevertheless, in the parameterized setting, one has to further check whether this translation 'respects' the parameter value of the two instances. In our analysis, this concerns parameters arity and $|\Phi|$ because these are the only two parameters that change when we replace a dependence atom with an equivalent independence atom. Recall that a dependence atom $\mathsf{dep}(\boldsymbol{x}; \boldsymbol{y})$ has arity $|\boldsymbol{x}|$, whereas, the equivalent independence atom $\boldsymbol{y} \perp_{\boldsymbol{x}} \boldsymbol{y}$ has arity $|\boldsymbol{x} \cup \boldsymbol{y}|$. In general one assumes that only dependence atoms of the form $\mathsf{dep}(\mathbf{x}; y)$ can appear in a $\mathcal{FO}(\mathsf{dep})$-formula which increases the arity by one. However, we do not restrict ourself to these atoms and prove that the reductions presented for the hardness of $\mathcal{FO}(\mathsf{dep})$ when parameterised by arity and $|\Phi|$ can be easily adapted to the case of $\mathcal{FO}(\perp)$. For arity, in the given reductions we will argue that replacing every dependence atom by independence atoms increases the arity only by a constant factor. For $|\Phi|$, we use the following observation.

▶ Remark 19. Let Φ be a $\mathcal{FO}(\mathsf{dep})$-formula and Φ' be the $\mathcal{FO}(\perp)$-formula obtained after replacing every dependence atom by an independence atom. Then, for any reasonable encoding of formulas we have that $|\Phi'| \leq \#\mathsf{atoms} \cdot |\Phi|^2$, where $\#\mathsf{atoms}$ denotes the number of dependence atoms in Φ and $\#\mathsf{atoms}(\Phi) \leq |\Phi|$.

That is, replacing a dependence atom $\mathsf{dep}(\boldsymbol{x}; \boldsymbol{y})$ by an independence atom $\boldsymbol{y} \perp_{\boldsymbol{x}} \boldsymbol{y}$ in Φ increases the size by $|\mathbf{y}| \leq |\Phi|$. Consequently, we have $|\Phi'| \leq |\Phi|^3$, and the hardness results for $\mathcal{FO}(\perp)$ when parameterized by $|\Phi|$ follow from the corresponding cases for $\mathcal{FO}(\mathsf{dep})$.

3.1 Data complexity (dc)

Classically, the data complexity of model checking for a fixed $\mathcal{C}$-formula Φ is **NP**-complete [**?**, **?**].

$x =$ 'variable'	$y =$ 'parity'	$u =$ 'clause'	$v =$ 'position'
p_1	1	1	0
p_2	0	1	1
p_3	0	1	2

Table 4 An example team for $(p_1 \vee \neg p_2 \vee \neg p_3)$

▶ **Proposition 20.** *For a fixed $\mathcal{C}$-formula, the problem whether an input structure $\mathcal{A}$ and a team T satisfies the formula is* **NP***-complete. That is, the data complexity of dependence and independence logic is* **NP***-complete.*

In this section we prove that none of the considered parameter lowers this complexity, except $|\mathcal{A}|$. The proof relies on the fact that the complexity of model checking for already a very simple formula (see below) is **NP**-complete.

▶ **Lemma 21.** *Let $k \in \{\#\mathsf{splits}, \#\mathsf{free\text{-}variables}, \#\mathsf{variables}, \#\forall, \mathsf{arity}, \mathsf{tw}(\mathcal{A})\}$. Then the problem k-$\mathsf{dc}(\mathcal{C})$, is* **paraNP***-c.*

Proof. The upper bound follows from Proposition 19. Kontinen [**?**, Theorem 4.9] proves that the data complexity for a fixed $\mathcal{FO}(\mathsf{dep})$-formula of the form $\mathsf{dep}(x;y) \vee \mathsf{dep}(u;v) \vee \mathsf{dep}(u;v)$ is already **NP**-complete. For clearity, we briefly sketch the reduction presented by Kontinen [**?**]. Let $\phi = \bigwedge_{i \leq m} (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ be an instance of 3-SAT. Consider the structure $\mathcal{A}$ over the empty vocabulary, that is, $\tau = \emptyset$. Let $A = \mathrm{Var}(\phi) \cup \{0, 1, \ldots, m\}$. The team T is constructed over variables $\{x, y, u, v\}$ that take values from A. As an example, the clause $(p_1 \vee \neg p_2 \vee \neg p_3)$ gives rise to assignments in Table 4. Notice that, a truth assignment θ for ϕ is constructed using the division of T according to each split. That is, $T \models \mathsf{dep}(x;y) \vee \mathsf{dep}(u;v) \vee \mathsf{dep}(u;v)$ if and only if $\exists P_0, P_1, P_2$ such that $\cup_i P_i = T$ for $i \leq 2$ and each P_i satisfies ith dependence atom. Let P_0 be such that $P_0 \models \mathsf{dep}(x;y)$, then we let $\theta(p_j) = 1 \iff \exists s \in P$, s.t. $s(x) = p_j$ and $s(y) = 1$. That is, one literal in each clause must be chosen in such a way that satisfies this clause, whereas, the remaining two literals per each clause are allowed to take values that does not satisfy it. As a consequence, each clause is satisfied by the variables chosen in this way, which proves correctness.

This implies that the 2-slice (for $\#\mathsf{splits}$-dc), 4-slice (for $\#\mathsf{free\text{-}variables}$-$\mathsf{dc}$ as well as $\#\mathsf{variables}$-dc), 0-slice (for $\#\forall$-dc), and 1-slice (for arity-dc) are **NP**-complete. Moreover, replacing each dependence atom in $\mathsf{dep}(x;y) \vee \mathsf{dep}(u;v) \vee \mathsf{dep}(u;v)$ by the equivalent independence atom increases the arity of independence atoms by at most 1. Consequently, the **paraNP**-hardness of these cases follow. Finally, the case for $\mathsf{tw}(\mathcal{A})$ also follows due to the reason that the vocabulary of the reduced structure is empty. As a consequence, our definition 7 yields a tree decomposition of width 1 trivially as no elements of the universe are related. This completes the proof to our lemma. ◀

▶ Remark 22. Recall that $|\Phi|$ as a parameter for $\mathsf{dc}(\mathcal{C})$ does not make sense as the input consists of $\langle \mathcal{A}, T \rangle$. That is, the formula Φ is already fixed which is stronger than fixing the size of Φ.

We now prove the only tractable case for the data complexity.

▶ **Lemma 23.** $|\mathcal{A}|$-$\mathsf{dc}(\mathcal{C}) \in$ **FPT**.

Proof. Notice first that restricting the universe size $|\mathcal{A}|$ polynomially bounds the teamsize $|T|$, due to Lemma 9 (for $\mathcal{FO}(\mathsf{dep})$) and Corollary 15 (for $\mathcal{FO}(\perp)$). This implies that the size of whole input is (polynomially) bounded by the parameter $|\mathcal{A}|$. The result follows trivially because any PP P is **FPT** when the input size is bounded by the parameter [**?**]. ◀

▶ **Lemma 24.** $|T|$-dc *is* **paraNP**-*complete.*

Proof. For a fixed sentence $\Phi \in \mathcal{FO}(\mathsf{dep})$ (that is, with no free variables) and for all models $\mathcal{A}$ and team T we have that $(\mathcal{A}, T) \models \Phi \iff (\mathcal{A}, \{\emptyset\}) \models \Phi$. As a result, the problem $\leq^{\mathbf{FPT}}$-reduces to the model checking problem with $|T| = 1$. Consequently, 1-slice of $|T|$-dc($\mathcal{FO}(\mathsf{dep})$) is **NP**-complete because model checking for a fixed $\mathcal{FO}(\mathsf{dep})$-sentence is also **NP**-complete [**?**]. This gives **paraNP**-hardness for $\mathcal{FO}(\mathsf{dep})$. The hardness for $\mathcal{FO}(\perp)$ uses Observation 17 additionally.

For the membership, note that given a structure $\mathcal{A}$ and a team T then for a fixed $\mathcal{C}$-formula Φ the question whether $(\mathcal{A}, T) \models \Phi$ is in **NP**. Consequently, giving **paraNP**-membership. ◀

A comparison with the propositional dependence ($\mathcal{PDL}$) and independence logic ($\mathcal{PIND}$) at this point might be interesting. If the formula size is a parameter then the model checking for $\mathcal{PDL}$ and $\mathcal{PIND}$ can be solved in **FPT**-time [**?**, **?**]. However, this is not the case for $\mathcal{FO}(\mathsf{dep})$ and $\mathcal{FO}(\perp)$ even if the formula is fixed in advance.

3.2 Expression and Combined Complexity (ec, cc)

Now we turn towards the expression and combined complexity of model checking for $\mathcal{C}$. Here again, in most cases the problem is still intractable for the combined complexity. However, expression complexity when parameterized by the formula size ($|\Phi|$) and the total number of variables (#variables) yields membership in **FPT**. Similar to the previous section, we first present results that directly translate from the known reductions for proving the **NEXP**-completeness for $\mathcal{C}$.

▶ **Lemma 25.** *Let* $k \in \{\, |\mathcal{A}|, \mathsf{tw}(\mathcal{A}), |T|, \#\text{free-variables} \,\}$. *Then both* k-cc($\mathcal{C}$) *and* k-ec($\mathcal{C}$) *are* **paraNEXP**-*complete.*

Proof. In the classical setting, **NEXP**-completeness of the expression and the combined complexity for $\mathcal{C}$ was shown by Grädel [**?**, Theorems 5.1 & 5.2]. This immediately gives membership in **paraNEXP**. Interestingly, for hardness the universe in the reduction consists of $\{0, 1\}$ with empty vocabulary and the formula obtained is a $\mathcal{FO}(\mathsf{dep})$-sentence. This implies that 2-slice (for $|\mathcal{A}|$), 1-slice (for $\mathsf{tw}(\mathcal{A})$), 1-slice (for $|T|$), and 0-slice (for the number of free variables) are **NEXP**-complete. As a consequence, **paraNEXP**-hardness for the mentioned cases follows for $\mathcal{FO}(\mathsf{dep})$. The corresponding cases for $\mathcal{FO}(\perp)$ also follow due to Observation 17 and this completes the proof. ◀

For the number of splits as a parameterization, we only know that this is also highly intractable, with the precise complexity open for now.

▶ **Lemma 26.** #splits-ec($\mathcal{C}$) *and* #splits-cc($\mathcal{C}$) *are both* **paraPSPACE**-hard.

Proof. Consider the equivalence of $\{\exists, \forall, \wedge\}$-$\mathcal{FO}$-MC to quantified constraint satisfaction problem (QCSP) [**?**, p. 418]. That is, the fragment of $\mathcal{FO}$ with only operations in $\{\exists, \forall, \wedge\}$ allowed. Then QCSP asks, whether the conjunction of quantified constraints ($\mathcal{FO}$-relations) is true in a fixed $\mathcal{FO}$-structure $\mathcal{A}$. This implies that already in the absence of a split operator (even when there are no dependence atoms), the model checking problem is **PSPACE**-hard. Consequently, the mentioned results follow. ◀

The formula size as a parameter presents varying behaviour depending upon if we consider the expression or the combined complexity. However, the complexity remains same for both logics we considered.

▶ **Lemma 27.** $|\Phi|$-cc$(\mathcal{C})$ *is* **paraNP***-complete.*

Proof. Notice that, due to Lemma 9, the size k of a formula Φ also bounds the maximum number of free variables in any subformula of Φ. This gives the membership in conjunction with [**?**, Theorem 5.1]. That is, the combined complexity of $\mathcal{C}$ is **NP**-complete if maximum number of free variables in any subformuala of Φ is fixed. The lower bound follows because of the construction by Kontinen [**?**] (see also Lemma 20) since for a fixed formula (of fixed size), the problem is already **NP**-complete. ◀

▶ **Lemma 28.** $|\Phi|$-ec$(\mathcal{C})$ *is in* **FPT***.*

Proof. Recall that in expression complexity, the team T and the structure $\mathcal{A}$ are fixed. Whereas, the size of the input formula Φ is a parameter. The result follows trivially because any PP P is **FPT** when the input size is bounded by the parameter. ◀

The expression complexity of $\mathcal{C}$ regarding the number of universal quantifiers as a parameter drops down to **paraNP**-completeness, which is still intractable but much lower than **paraNEXP**-completeness. However, regarding the combined complexity we can only prove the membership in **XNP**, with **paraNP**-lower bound.

▶ **Lemma 29.** $\#\forall$-ec$(\mathcal{C})$ *is* **paraNP***-complete.*

Proof. We first prove the lower bound for $\#\forall$-ec$(\mathcal{FO}(\mathsf{dep}))$ through a reduction form the satisfiability problem for propositional dependence logic ($\mathcal{PDL}$). That is, given a $\mathcal{PDL}$-formula ϕ, whether there is a team T such that $T \models \phi$? Let ϕ be a $\mathcal{PDL}$-formula over propositional variables $p_1, \dots, p_n$. For $i \leq n$, let x_i denote a variable corresponding to the proposition p_i. Let $\mathcal{A} = \{0, 1\}$ be the structure over empty vocabulary. Clearly ϕ is satisfiable iff $\exists p_1 \dots \exists p_n \phi$ is satisfiable iff $(\mathcal{A}, \{\emptyset\}) \models \exists x_1 \dots \exists x_n \phi'$, where ϕ' is a $\mathcal{FO}(\mathsf{dep})$-formula obtained from ϕ by simply replacing each proposition p_i by the variable x_i. Notice that the reduced formula does not have any universal quantifier, that is $\#\forall(\phi') = 0$. This gives **paraNP**-hardness of $\#\forall$-ec$(\mathcal{FO}(\mathsf{dep}))$ since the satisfiability for $\mathcal{PDL}$ is **NP**-complete [**?**]. Moreover, the hardness of $\#\forall$-ec$(\mathcal{FO}(\perp))$ also follows due to Observation 17.

For membership, notice first that a $\mathcal{FO}(\mathsf{dep})$-sentence Φ with k universal quantifiers can be reduced in **P**-time to an $\mathcal{ESO}$-sentence Ψ of the form $\exists f_1 \dots \exists f_r \forall x_1 \dots \forall x_k \psi$ [**?**, Cor. 3.9], where ψ is a quantifier free $\mathcal{FO}$-formula, $r \in \mathbb{N}$, and each function symbol f_i is at most k-ary for $1 \leq i \leq r$. Finally, $(\mathcal{A}, \{\emptyset\}) \models \Phi \iff \mathcal{A} \models \bigvee_{f_1} \dots \bigvee_{f_r} \forall x_1 \dots \forall x_k \psi'$. Where the latter question can be solved by guessing an interpretation for each function symbol f_i and $i \leq r$. This requires $r \cdot |\mathcal{A}|^k$ guessing steps, and can be achieved in **paraNP**-time for a fixed structure $\mathcal{A}$ (as we consider expression complexity). Similarly, an $\mathcal{FO}(\perp)$sentence Φ with k universal quantifiers can be reduced in **P**-time to an $\mathcal{ESO}$-sentence Ψ of the form $\exists f_1 \dots \exists f_r \forall x_1 \dots \forall x_k \forall x_{k+1} \psi$ [**?**, Proposition 20]. The only difference being an additional universal quantifier in the case of $\mathcal{FO}(\perp)$-sentences. It is worth mentioning that the proof by Kontinen and Hannula [**?**, Proposition 20] does not state explicitly that the function symbols can be assumed to have arity at most k. However, this can be assumed using a result by Durand et al. [**?**, Theorem 5.11]. Consequently, the membership in **paraNP** follows for $\#\forall$-ec$(\mathcal{C})$.

Notice that the arity of function symbols in the **paraNP**-membership above is bounded by k if Φ is a $\mathcal{C}$-sentence. However, if Φ is a $\mathcal{C}$-formulas with m free variables then the arity of function symbols as well as the number of universal quantifiers in the reduction, both are bounded by $k+m$ where $k = \#\forall(\Phi)$ and $m = \#\mathsf{free\text{-}variables}(\Phi)$. Nevertheless, recall that for ec, the team is also fixed. Moreover, due to Lemma 9 the collection of free variables in Φ has constant size. This implies that the reduction above provides an $\mathcal{ESO}$-sentence with $k+m$ universal quantifiers as well as function symbols of arity $k+m$ at most. Finally, guessing the interpretation for functions still takes **paraNP**-steps (because m is constant) and consequently, we get **paraNP**-membership for open $\mathcal{C}$-formulas as well. ◀

The following corollary immediately follows from the proof above.

▶ **Corollary 30.** $(\#\forall + \#\mathsf{free\text{-}variables})\text{-}\mathsf{ec}(\mathcal{C})$ *is* **paraNP**-*complete.*

▶ **Lemma 31.** $\#\forall\text{-}\mathsf{cc}(\mathcal{C})$ *is* **paraNP**-*hard. Moreover,* $\#\forall\text{-}\mathsf{cc}(\mathcal{C})$ *is in* **XNP** *for* $\mathcal{C}$*-sentences.*

Proof. The **paraNP**-lower bound follows due to the fact that the expression complexity of $\mathcal{C}$ is already **paraNP**-complete when parameterized by $\#\forall$ (Lemma 28).

For sentences, similar to the proof in Lemma 28, a $\mathcal{C}$-sentence Φ can be translated to an equivalent $\mathcal{ESO}$-sentence Ψ in polynomial time. However, if the structure is not fixed as for expression complexity, then the computation of interpretations for functions can no longer be done in **paraNP**-time, but requires non-deterministic $|\mathcal{A}|^k$-time for each guessed function, where $k = \#\forall$. Consequently, we reach only membership in **XNP** for sentences. ◀

For open formulas, we do not know if $\#\forall\text{-}\mathsf{cc}(\mathcal{C})$ is also in **XNP**. Our proof technique does not immediately settle this case as the team is not fixed for cc.

Similar to the case of universal quantifiers, the arity as a parameter also reduces the complexity for both logics, but not as much as the universal quantifiers. Moreover, the precise combined complexity when parameterized by the arity is also open.

▶ **Lemma 32.** $\mathsf{arity}\text{-}\mathsf{ec}(\mathcal{C})$ *is* **paraPSPACE**-*complete.*

Proof. For hardness, notice that the expression complexity of $\mathcal{FO}$ is **PSPACE**-complete. This implies that already in the absence of any (in)dependence atoms, the complexity remains **PSPACE**-hard, as a consequence, the 0-slice of $\mathsf{arity}\text{-}\mathsf{ec}(\mathcal{C})$ is **PSPACE**-hard.

For membership, notice that a $\mathcal{FO}(\mathsf{dep})$-sentence Φ with k-ary dependence atoms can be reduced in **P**-time to an $\mathcal{ESO}$-sentence Ψ of the form $\exists f_1 \dots \exists f_r \psi$ [**?**, Thm. 3.3], where ψ is an $\mathcal{FO}$-formula and each function symbol f_i is at most k-ary for $1 \le i \le r$. Finally, $\mathcal{A} \models \Phi \iff \mathcal{A} \models \bigvee_{f_1} \dots \bigvee_{f_r} \psi'$. That is, one needs to guess the interpretation for each function symbol f_i, which can be done in **paraNP**-time. Finally, evaluating an $\mathcal{FO}$-formula ψ' for a fixed structure $\mathcal{A}$ can be done in **PSPACE**-time. This yields membership in **paraPSPACE**. Moreover, if Φ is an open $\mathcal{FO}(\mathsf{dep})$-formula then the result follows due to a similar discussion as in the proof of Lemma 28. Finally, for $\mathcal{FO}(\perp)$ the result follows because $\mathcal{FO}(\perp)(k\text{-}ind) = \mathcal{FO}(\mathsf{dep})(k\text{-}dep)$ [**?**, Theorem 35] . That is, the fragment of independence logic obtained by allowing only k-ary independence atoms is equivalent to the fragment of dependence logic obtained by allowing only k-ary dependence atoms. This proves the desired result. ◀

The combination $(\mathsf{arity} + \#\mathsf{free\text{-}variables})$ also does not lower the expression complexity as discussed before in the case of $\#\forall$.

▶ **Corollary 33.** $(\mathsf{arity} + \#\mathsf{free\text{-}variables})\text{-}\mathsf{ec}(\mathcal{C})$ *is* **paraPSPACE**-*complete.*

▶ **Lemma 34.** arity-cc($\mathcal{C}$) *is* **paraPSPACE**-*hard.*

Proof. Consider the fragment of $\mathcal{FO}(\mathsf{dep})$ with only dependence atoms of the form $\mathsf{dep}(;x)$, the so-called constancy logic. The combined complexity of constancy logic is **PSPACE**-complete [**?**, Theorem 5.3]. This implies that the 0-slice of arity-cc($\mathcal{FO}(\mathsf{dep})$) is **PSPACE**-hard, proving the result. The hardness for $\mathcal{FO}(\perp)$ follows because of the equivalence $\mathsf{dep}(;x) \equiv x \perp_{\emptyset} x$. ◀

The combined complexity of model checking for constancy logic is **PSPACE** [**?**, Thm. 5.3]. Aiming for an **paraPSPACE**-upper bound via squeezing the fixed arity of dependence atoms (in some way) into constancy atoms is unlikely to happen as $\mathcal{FO}(\mathsf{dep})$ (as well as $\mathcal{FO}(\perp)$) captures $\mathcal{ESO}$ whereas constancy logic for sentences (and also open formulas) collapses to $\mathcal{FO}$ [**?**].

Notice that a similar reduction as in the proof of Lemma 28 holds from $\mathcal{PL}$, in which both parameters (#∀ and arity) are bounded. This implies that there is no hope for tractability even when both parameters are considered together. That is, the expression complexity remains **paraNP**-complete when parameterized by the combination of parameters (#∀, arity).

▶ **Corollary 35.** (#∀ + arity)-ec($\mathcal{C}$) *is also* **paraNP**-*complete.*

Finally, for the parameter total number of variables, the expression complexity drops to **FPT** whereas, the combined complexity drops to **paraNP**-completeness. The case of expression complexity is particularly interesting. This is due to the reason that it was posed as an open question by Virtema [**?**] whether the expression complexity of the fixed variable fragment of dependence logic ($\mathcal{FO}(\mathsf{dep})^k$) is **NP**-complete similar to the case of the combined complexity therein. We answer this negatively by stating **FPT**-membership for #variables-ec, which as a corollary proves that the expression complexity of $\mathcal{FO}(\mathsf{dep})^k$ is in **P** for each $k \geq 1$.

▶ **Lemma 36.** #variables-cc($\mathcal{C}$) *is* **paraNP**-*complete.*

Proof. Notice that if the total number of variables in a $\mathcal{C}$-formula Φ is fixed, then the number of free variables in any subformula ψ of Φ is also fixed. This implies the membership in **paraNP** due to [**?**, Theorem 5.1]. On the other hand, by [**?**, Theorem 3.9.6] we know that the combined complexity of $\mathcal{D}^k$ is **NP**-complete. This implies that for each k, the k-slice of the problem is **NP**-hard. The desired hardness for $\mathcal{FO}(\perp)^k$ follows due to Observation 17. This gives the lower bounds for both logics. ◀

The following lemma once again utilizes the fact that a $\mathcal{C}$-formula can be reduced to an equivalent $\mathcal{ESO}$-formula. However, an important observation here is that this reduction also preserves the number of variables in the formula

▶ **Lemma 37.** #variables-ec($\mathcal{C}$) *is* **FPT**.

Proof. Given a formula Φ of dependence logic with k variables, we can construct an equivalent formula Ψ of $\mathcal{ESO}^{k+1}$ in polynomial time [**?**, Theorem 3.3.17]. Moreover, since the structure $\mathcal{A}$ is fixed, there exists a reduction of Ψ to an $\mathcal{FO}$-formula ψ with $k+1$ variables (big disjunction on the universe elements for each second order existential quantifier). Finally, the model checking for $\mathcal{FO}$-formulas with k variables is solvable in time $O(|\psi| \cdot |A|^k)$ [**?**, Prop 6.6]. This implies the membership in **FPT**. If Φ is an $\mathcal{FO}(\perp)$-formula, then the (worst case) reduction of Φ into an $\mathcal{ESO}$-sentence uses $3k$ variables [**?**, Prop. 14]. The above discussion gives a similar **FPT**-algorithm for $\mathcal{FO}(\perp)$ running in time $O(|\psi| \cdot |A|^k)$. ◀

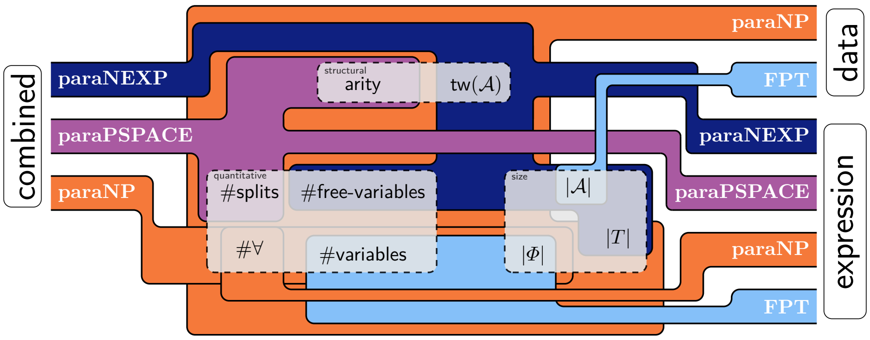

Figure 3 Complexity classification overview for model checking problem of (in)dependence logic, that takes grouping of parameters (quantitative, size, structural) and complexity classes into account.

▶ **Corollary 38.** *The expression complexity of* $\mathcal{FO}(\mathsf{dep})^k$ *is in* **P** *for every* $k \geq 1$.

Proof. Since both, the number of variables and the universe size is fixed. The runtime of the form $O(|\psi| \cdot |A|^k)$ in Lemma 36 implies membership in **P**. ◀

4 Conclusion

In this paper, we analyzed the parameterized complexity classification of model checking for dependence ($\mathcal{FO}(\mathsf{dep})$) and independence logic ($\mathcal{FO}(\perp)$) with respect to nine different parameters (see Table 2 for an overview of the results). In Fig. 3 we depict a different kind of presentation of our results that also takes the grouping of parameters into quantitative, size related, and structural into account. Interestingly, the complexity for both considered logics remains same under each parameterization. Moreover, the complexity of $\mathcal{FO}(\perp)$ also remains same under both (strict and lax) semantics . The data complexity of $\mathcal{C}$ shows a dichotomy (**FPT** vs. **paraNP**-complete), where surprisingly there is only one case ($|\mathcal{A}|$) where one can reach **FPT**. This is even more surprising in the light of the fact that the expression (ec and the combined (cc) complexities under the same parameter are still highly intractable. Furthermore, there are parameters when cc and ec vary in the complexity (#variables). The combined complexity of $\mathcal{C}$ stays intractable under any of the investigated parameterizations. It might be interesting to study combination of parameters and see their joint effect on the complexity (yet, Corollaries 29, 32, 34 tackle already some cases).

We want to close this presentation with some further questions:

- What other parameters could be meaningful (e.g., number of conjunction, number of existential quantifiers, treewidth of the formula)?
- What is the exact complexity of #∀-cc($\mathcal{C}$), #splits-ec($\mathcal{C}$)/-cc($\mathcal{C}$), arity-cc($\mathcal{C}$)?
- What new insights brings the parameterized complexity analysis for inclusion logic?