# Parameterized Complexity of Propositional Inclusion and Independence Logic

Yasir Mahmood[1][0000−0002−5651−5391] and Jonni Virtema[2][0000−0002−1582−3718]

[1] DICE group, Department of Computer Science, Paderborn University, Germany
yasir.mahmood@uni-paderborn.de
[2] Department of Computer Science, University of Sheffield, United Kingdom
j.t.virtema@sheffield.ac.uk

**Abstract.** We give a comprehensive account on the parameterized complexity of model checking and satisfiability of propositional inclusion and independence logic. We discover that for most parameterizations the problems are either in FPT or paraNP-complete.

**Keywords:** Propositional Logic · Team Semantics · Model checking · Satisfiability · Parameterized Complexity

## 1 Introduction

The research program on team semantics was conceived in the early 2000s to create a unified framework to study logical foundations of different notions of dependence between variables. Soon after the introduction of first-order dependence logic [28], the framework was extended to cover propositional and modal logic [29]. In this context, a significant step was taken in [5], where the focus shifted to study dependencies between formulas instead of variables. The framework of team semantics has been proven to be remarkably malleable. During the past decade the framework has been re-adapted for the needs of an array of disciplines. In addition to the modal variant, team semantics has been generalized to temporal [19] and probabilistic [4] frameworks, and fascinating connections to fields such as database theory [11], statistics [1], real valued computation [9], verification [20], and quantum information theory [16] have been identified.

Boolean satisfiability problem (SAT) and quantified Boolean formula problem (QBF) have had a widespread influence in diverse research communities. In particular, QBF solving techniques are important in application domains such as planning, program synthesis and verification, adversary games, and non-monotonic reasoning, to name a few [27]. Further generalizations of QBF are the dependency quantified Boolean formula problem (DQBF) and alternating DQBF which allow richer forms of variable dependence [10,24,25]. Propositional logics with team semantics offer a fresh perspective to study enrichments of SAT and QBF. Indeed, the so-called *propositional dependence logic* ($\mathcal{PDL}$) is known to coincide with DQBF, whereas quantified propositional logics with team semantics have a close connection to alternating DQBF [10,30].

| Instructor | Time | Room | Course | Responsible |
|---|---|---|---|---|
| Antti | 09:00 | A.10 | Genetics | Antti |
| Antti | 11:00 | A.10 | Chemistry | Juha |
| Antti | 15:00 | B.20 | Ecology | Antti |
| Jonni | 10:00 | C.30 | Bio-LAB | Jonni |
| Juha | 10:00 | C.30 | Bio-LAB | Jonni |
| Juha | 13:00 | A.10 | Chemistry | Juha |

| $i_1 i_2$ | $t_1 t_2 t_3$ | $r_1 r_2$ | $c_1 c_2$ | $p_1 p_2$ |
|---|---|---|---|---|
| 00 | 110 | 11 | 11 | 00 |
| 00 | 111 | 11 | 00 | 10 |
| 00 | 000 | 00 | 01 | 00 |
| 01 | 001 | 01 | 10 | 01 |
| 10 | 001 | 01 | 10 | 01 |
| 10 | 010 | 11 | 00 | 10 |

**Table 1.** (Left) An example database with 5 attributes and universe size 15. (Right) An encoding with $3 \cdot \lceil \log_2(3) \rceil + \lceil \log_2(5) \rceil + \lceil \log_2(4) \rceil$ many propositional variables.

Propositional dependency logics extend propositional logic with atomic dependency statements describing various forms of variable dependence. In this setting, formulas are evaluated over propositional teams (i.e, sets of propositional assignments with common variable domain). An *inclusion atom* $\boldsymbol{x} \subseteq \boldsymbol{y}$ is true in a team $T$, if $\forall s \in T \, \exists t \in T$ such that $s(\boldsymbol{x}) = t(\boldsymbol{y})$. An *independence atom* $\boldsymbol{x} \perp_{\boldsymbol{z}} \boldsymbol{y}$ expresses that in a team $T$, for any fixed value for the variables in $\boldsymbol{z}$ the values for $\boldsymbol{x}$ and $\boldsymbol{y}$ are informationally independent. The extension of propositional logic with inclusion and independence atoms yield inclusion ($\mathcal{PINC}$) and independence ($\mathcal{PIND}$) logics, respectively.

*Example 1.* Table 1 illustrates an example from relational databases. The set of records corresponds to a team, that satisfies the dependency $\texttt{Responsible} \subseteq \texttt{Instructor}$. Moreover, it violates the independence $\texttt{Instructor} \perp_{\texttt{Course}} \texttt{Time}$ as witnessed by tuples (Antti, 11:00, A.10, Chemistry, Juha) and (Juha, 13:00, A.10, Chemistry, Juha). In propositional logic setting, datavalues can be represented as bit strings of appropriate length (as depicted in Table 1).

The complexity landscape of the classical (non-parameterized) decision problems — satisfiability, validity, and model checking — is well mapped in the propositional and modal team semantics setting (see [14, page 627] for an overview). Parameterized complexity theory, pioneered by Downey and Fellows [2], is a widely studied subarea of complexity theory. The motivation being that it provides a deeper analysis than the classical complexity theory by providing further insights into the source of intractability. The idea here is to identify meaningful parameters of inputs such that fixing those makes the problem tractable. One example of a fruitful parameter is the treewidth of a graph. A parameterized problem (PP) is called fixed parameter tractable, or in **FPT** for short, if for a given input $x$ with parameter $k$, the membership of $x$ in PP can be decided in time $f(k) \cdot p(|x|)$ for some computable function $f$ and polynomial $p$. That is, for each fixed value of $k$ the problem is tractable in the classical sense of tractability (in **P**), and the degree of the polynomial is independent of the parameter. The class **paraNP** consists of problems decidable in time $f(k) \cdot p(|x|)$ on a non-deterministic machine.

In the propositional team semantics setting, the study of parameterized complexity was initiated by Meier and Reinbold [23] in the context of parameter-

| | $\mathcal{PIND}$ | | $\mathcal{PINC}$ | |
|---|---|---|---|---|
| Parameter | MC | SAT | $MC_s$ | SAT |
| formula-tw | **paraNP**[18] | **FPT**[19] | **paraNP**[17] | in **paraNP**[13] |
| formula-team-tw | **FPT**[8] | - | **FPT**[8] | - |
| team-size | **FPT**[7] | - | **FPT**[7] | - |
| formula-size | **FPT**[8] | Trivial | **FPT**[8] | Trivial |
| formula-depth | **FPT**[8] | **FPT**[9] | **FPT**[8] | **FPT**[9] |
| #variables | **FPT**[8] | **FPT**[9] | **FPT**[8] | **FPT**[9] |
| #splits | **paraNP**[18] | **FPT**[19] | **paraNP**[17] | **P**[14] if #splits=0 |
| arity | **paraNP**[18] | **paraNP**[19] | **paraNP**[17] | **paraNP**[10] |

**Table 2.** Overview of parameterized complexity results with pointers to the results. The **paraNP**-cases are complete, except for only membership in the first row. $MC_s$ denotes model checking for strict semantics whereas MC/SAT refer to both semantics.

ized enumeration problems, and by Mahmood and Meier [22] in the context of classical decision problems, for $\mathcal{PDL}$. In the first-order team semantics setting, Kontinen et al. [18] studied parameterized model checking of dependence and independence logic, and in [17] introduced the *weighted-definability* problem for dependence, inclusion and independence logic thereby establishing a connection with the parameterized complexity classes in the well-known **W**-*hierarchy*.

We focus on the parameterized complexity of model checking (MC) and satisfiability (SAT) of propositional inclusion and independence logic. We consider both *lax* and *strict* semantics. The former is the prevailing semantics in the team semantics literature. The past rejection of strict semantics was based on the fact that it does not satisfy locality [8] (the locality principle dictates that satisfaction of a formula should be invariant on the truth values of variables that do not occur in the formula). Recent works have revealed that locality of strict semantics can be recovered by moving to multiteam semantics (here teams are multisets) [3]. Since, in propositional team semantics, the shift from teams to multiteams has no complexity theoretic implications, we stick with the simpler set based semantics for our logics. In the model checking problem, one is given a team $T$ and a formula $\phi$, and the task is to determine whether $T \models \phi$. In the satisfiability problem, one is given a formula $\phi$, and the task is to decide whether there exists a non-empty satisfying team $T$ for $\phi$. Table 2 gives an overview of our results. We consider only strict semantics for MC of $\mathcal{PINC}$, since for lax semantics the problem is tractable already in the non-parameterized setting [14, Theorem 3.5].

## 2 Preliminaries

We assume familiarity with standard notions in complexity theory such as classes **P**, **NP** and **EXP** [26]. We give a short exposition of relevant concepts from parameterized complexity theory. For a broader introduction consider the textbook of Downey and Fellows [2], or that of Flum and Grohe [7].

A *parameterized problem* (PP) $\Pi \subseteq \Sigma^* \times \mathbb{N}$ consists of tuples $(x, k)$, where $x$ is called an instance and $k$ is the (value of the) parameter.

**FPT and paraNP.** Let $\Pi$ be a PP over $\Sigma^* \times \mathbb{N}$. Then $\Pi$ is *fixed parameter tractable* (**FPT** for short) if it can be decided by a deterministic algorithm $\mathcal{A}$ in time $f(k) \cdot p(|x|)$ for any input $(x, k)$, where $f$ is a computable function and $p$ is a polynomial. If the algorithm $\mathcal{A}$ is non-deterministic instead, then $\Pi$ belongs to the class **paraNP**.

The notion of hardness in parameterized setting is employed by fpt-reductions.

**fpt-reductions.** Let $\Pi \subseteq \Sigma^* \times \mathbb{N}$ and $\Theta \subseteq \Gamma^* \times \mathbb{N}$ be two PPs. Then $\Pi$ is *fpt-reducible* to $\Theta$, if there exists an fpt-computable function $g\colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$ such that (1) for all $(x, k) \in \Sigma^* \times \mathbb{N}$ we have that $(x, k) \in \Pi \Leftrightarrow g(x, k) \in \Theta$ and (2) there exists a computable function $h\colon \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ and $g(x, k) = (x', k')$ we have that $k' \leq h(k)$.

We will use the following result to prove **paraNP**-hardness. Let $\Pi$ be a PP over $\Sigma^* \times \mathbb{N}$. Then the *$\ell$-slice of $\Pi$*, for $\ell \geq 0$, is the set $\Pi_\ell := \{\, x \mid (x, \ell) \in \Pi \,\}$.

**Proposition 2 ([7, Theorem 2.14]).** *Let $\Pi$ be a PP in* **paraNP**. *If there exists an $\ell \geq 0$ such that $\Pi_\ell$ is* **NP***-complete, then $\Pi$ is* **paraNP**-complete.

Moreover, we will use the following folklore result to get several upper bounds.

**Proposition 3.** *Let $Q$ be a problem such that $(Q, k)$ is* **FPT** *and let $\ell$ be a parameter with $k \leq f(\ell)$ for some computable function $f$. Then $(Q, \ell)$ is* **FPT**.

**Propositional Team Based Logics.** Let Var be a countably infinite set of variables. The syntax of propositional logic ($\mathcal{PL}$) is defined via the following grammar: $\phi ::= x \mid \neg x \mid \phi \vee \phi \mid \phi \wedge \phi$, where $x \in \mathrm{Var}$. Observe that we allow only atomic negations. As usual $\top := x \vee \neg x$ and $\bot := x \wedge \neg x$. Propositional dependence logic $\mathcal{PDL}$ is obtained by extending $\mathcal{PL}$ by atomic formulas of the form $=(\boldsymbol{x}; \boldsymbol{y})$, where $\boldsymbol{x}, \boldsymbol{y} \subset \mathrm{Var}$ are finite tuples of variables. Similarly, adding inclusion atoms $\boldsymbol{x} \subseteq \boldsymbol{y}$ where $(|\boldsymbol{x}| = |\boldsymbol{y}|)$ and independence atoms $\boldsymbol{x} \bot_{\boldsymbol{z}} \boldsymbol{y}$ gives rise to propositional inclusion ($\mathcal{PINC}$) and independence ($\mathcal{PIND}$) logic, respectively. When we wish to talk about any of the three considered logics, we simply write $\mathcal{L}$. That is, unless otherwise stated, $\mathcal{L} \in \{\mathcal{PDL}, \mathcal{PINC}, \mathcal{PIND}\}$. For an assignment $s$ and a tuple $\boldsymbol{x} = (x_1, \ldots, x_n)$, $s(\boldsymbol{x})$ denotes $(s(x_1), \ldots, s(x_n))$.

**Team Semantics.** Let $\phi, \psi$ be $\mathcal{L}$-formulas and $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \subset \mathrm{Var}$ be finite tuples of variables. A *team* $T$ is a set of assignments $t\colon \mathrm{Var} \to \{\, 0, 1 \,\}$. The satisfaction relation $\models$ is defined as follows:

$$
\begin{array}{lll}
T \models x & \text{iff} & \forall t \in T : t(x) = 1, \\
T \models \neg x & \text{iff} & \forall t \in T : t(x) = 0, \\
T \models \phi \wedge \psi & \text{iff} & T \models \phi \text{ and } T \models \psi, \\
T \models \phi \vee \psi & \text{iff} & \exists T_1, T_2 (T_1 \cup T_2 = T) : T_1 \models \phi \text{ and } T_2 \models \psi, \\
T \models \boldsymbol{x} \subseteq \boldsymbol{y} & \text{iff} & \forall t \in T \, \exists t' \in T : t(\boldsymbol{x}) = t'(\boldsymbol{y}), \\
T \models \boldsymbol{x} \bot_{\boldsymbol{z}} \boldsymbol{y} & \text{iff} & \forall t, t' \in T : t(\boldsymbol{z}) = t'(\boldsymbol{z}), \exists t'' : t''(\boldsymbol{xzy}) = t(\boldsymbol{xz}) t'(\boldsymbol{y}).
\end{array}
$$

Intuitively, an inclusion atom $\boldsymbol{x} \subseteq \boldsymbol{y}$ is true if the value taken by $\boldsymbol{x}$ under an assignment $t$ is also taken by $\boldsymbol{y}$ under some assignment $t'$. Moreover, the independence atom $\boldsymbol{x}\perp_{\boldsymbol{z}}\boldsymbol{y}$ has the meaning that whenever the value for $\boldsymbol{z}$ is fixed under two assignments $t$ and $t'$, then there is an assignment $t''$ which maps $\boldsymbol{x}$ and $\boldsymbol{y}$ according to $t$ and $t'$, respectively. We can interpret the dependence atom $=(\boldsymbol{x};\boldsymbol{y})$ as the independence atom $\boldsymbol{y}\perp_{\boldsymbol{x}}\boldsymbol{y}$. The operator $\vee$ is also called a split-junction in the context of team semantics. Note that in the literature there exist two semantics for the split-junction: *lax* and *strict* semantics (e.g., Hella et al. [14]). Strict semantics requires the "splitting of the team" to be a partition whereas lax semantics allows an "overlapping" of the team. Regarding $\mathcal{PDL}$ and $\mathcal{PINC}$, the complexity for SAT and MC is the same irrespective of the considered semantics. However, the picture is different for MC in $\mathcal{PINC}$ as depicted in [14, page 627]. For any logic $\mathcal{L}$, we denote MC under strict (respectively, lax) semantics by $\mathrm{MC}_s(\mathrm{MC}_l)$. Moreover, $\mathrm{MC}_l$ is in $\mathbf{P}$ for $\mathcal{PINC}$ and consequently, we have only $\mathrm{MC}_s$ in Table 2.

## 3 Graph Representation of the Input

In order to consider specific structural parameters, we need to agree on a representation of an input instance. We follow the conventions given in [22]. Well-formed $\mathcal{L}$-formulas, for every $\mathcal{L} \in \{\mathcal{PDL}, \mathcal{PINC}, \mathcal{PIND}\}$, can be seen as binary trees (the syntax tree) with leaves as atomic subformulas (variables and dependency atoms). Similarly to $\mathcal{PDL}$ [22], we take the syntax structure (defined below) rather than syntax tree as a graph structure in order to consider treewidth as a parameter. We use the same graph representation for each logic $\mathcal{L}$. That is, when an atom $=(\boldsymbol{x};\boldsymbol{y})$ is replaced by either $\boldsymbol{x} \subseteq \boldsymbol{y}$ or $\boldsymbol{x}\perp_{\emptyset}\boldsymbol{y}$, the graph representation, and hence, the treewidth of this graph remains the same. Also, in the case of MC, we include assignments in the graph representation. In the latter case, we consider the Gaifman graph of the structure that models both, the team and the input formula.

**Syntax Structure.** Let $\phi$ be an $\mathcal{L}$-formula with propositions $\{x_1, \ldots, x_n\}$ and $T = \{ s_1, \ldots s_m \}$ a team. The *syntax structure* $\mathcal{A}_{T,\phi}$ has the vocabulary, $\tau_{T,\phi} := \{ \mathsf{VAR}^1, \mathsf{SF}^1, \succcurlyeq^2, \mathsf{DEP}^2, \mathsf{isTrue}^2, \mathsf{isFalse}^2, r, c_1, \ldots, c_m \}$, where the superscript denote the arity of each relation. The universe of $\mathcal{A}_{T,\phi}$ is $A := \mathrm{SF}(\phi) \cup \mathrm{Var}(\phi) \cup \{ c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}} \}$, where $\mathrm{SF}(\phi)$ and $\mathrm{Var}(\phi)$ denote the set of subformulas and variables appearing in $\phi$, respectively.

- $\mathsf{SF}$ and $\mathsf{VAR}$ are unary relations: 'is a subformula of $\phi$' and 'is a variable in $\phi$'.
- $\succcurlyeq$ is a binary relation such that $\psi \succcurlyeq^{\mathcal{A}} \alpha$ iff $\alpha$ is an immediate subformula of $\psi$. That is, either $\psi = \neg\alpha$ or there is a $\beta \in \mathrm{SF}(\phi)$ such that $\psi = \alpha \oplus \beta$ where $\oplus \in \{\wedge, \vee\}$. Moreover, $r$ is a constant symbol representing $\phi$.
- $\mathsf{DEP}$ is a binary relation connecting $\mathcal{L}$-atoms and its parameters. For example, if $\alpha = \boldsymbol{x} \subseteq \boldsymbol{y}$ and $x, y \in \boldsymbol{x} \cup \boldsymbol{y}$, then $\mathsf{DEP}(\alpha, x)$ and $\mathsf{DEP}(x, y)$ are true.
- The set $\{ c_1, \ldots, c_m \}$ encodes the team $T$. Each $c_i \in \tau_{T,\phi}$ corresponds to an assignment $s_i \in T$ for $i \leq m$, interpreted as $c_i^{\mathcal{A}} \in A$.

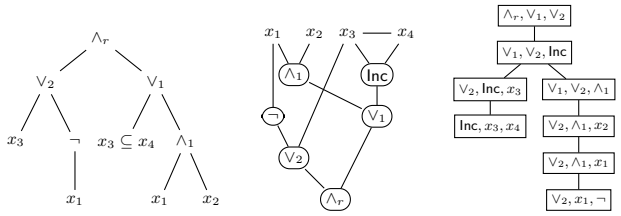



**Fig. 1.** An example syntax tree (left) with the corresponding Gaifman graph (middle) and a tree decomposition (right) for $(x_3 \vee \neg x_1) \wedge \big(x_3 \subseteq x_4 \vee (x_1 \wedge x_2)\big)$. We abbreviated subformulas in the inner vertices of the Gaifman graph for better presentation.

– isTrue and isFalse relate VAR with $c_1, \ldots, c_m$. isTrue$(c, x)$ (resp., isFalse$(c, x)$) is true iff $x$ is mapped to 1 (resp., 0) by the assignment in $T$ interpreted by $c$.

The *syntax structure* $\mathcal{A}_\phi$ over a vocabulary $\tau_\phi$ is defined analogously. Here $\tau_\phi$ neither contains the team related relations nor the constants $c_i^{\mathcal{A}}$ for $1 \leq i \leq m$.

**Gaifman graph.** Let $T$ be a team, $\phi$ an $\mathcal{L}$-formula, $\mathcal{A}_{T,\phi}$ and $A$ as above. The *Gaifman graph* $G_{T,\phi} = (A, E)$ of the $\tau_{T,\phi}$-structure $\mathcal{A}_{T,\phi}$ is defined as $E := \big\{\, \{u, v\} \;\big|\; u, v \in A, \text{ such that there is an } R \in \tau_{T,\phi} \text{ with } (u, v) \in R \,\big\}$. Analogously, we let $G_\phi$ to be the *Gaifman graph* for the $\tau_\phi$-structure $\mathcal{A}_\phi$.

Note that for $G_\phi$ we have $E = \mathsf{DEP} \cup \succcurlyeq$ and for $G_{T,\phi}$ we have that $E = \mathsf{DEP} \cup \succcurlyeq \cup \mathsf{isTrue} \cup \mathsf{isFalse}$.

**Treewidth.** A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (B, E_T)$, where the vertex set $B \subseteq \mathcal{P}(V)$ is a collection of *bags* and $E_T$ is the edge relation such that (1) $\bigcup_{b \in B} b = V$, (2) for every $\{\, u, v \,\} \in E$ there is a bag $b \in B$ with $u, v \in b$, and (3) for all $v \in V$ the restriction of $T$ to $v$ (the subset with all bags containing $v$) is connected. The *width* of a 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 graph $G$ is the minimum over widths of all tree decompositions of $G$. The treewidth of a tree is one. Intuitively, it measures the tree-likeness of a given graph.

*Example 4 (Adapted from [22]).* Figure 1 represents the Gaifman graph of the syntax structure $\mathcal{A}_\phi$ (in middle) with a tree decomposition (on the right). Since the largest bag is of size 3, the treewidth of the given decomposition is 2. Figure 2 presents the Gaifman graph of $\mathcal{A}_{T,\phi}$, that is, when the team $T = \{s_1, s_2\} = \{0011, 1110\}$ is part of the input (an assignment $s$ is denoted as $s(x_1 \ldots x_4)$).

**Parameterizations.** We consider eight different parameters for MC and six for SAT. For MC, we include formula-tw, formula-team-tw, team-size, formula-size, #variables, formula-depth, #splits and arity. However, for SAT, formula-team-tw and team-size are not meaningful. All these parameters arise naturally in problems we study. Let $T$ be a team and $\phi$ an $\mathcal{L}$-formula. #splits denotes the number of times a split-junction ($\vee$) appears in $\phi$ and #variables denotes the number of distinct propositional variables. formula-depth is the depth of the syntax tree

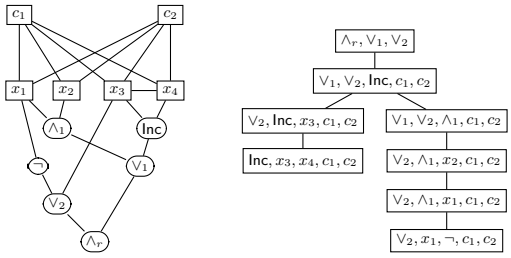



**Fig. 2.** The Gaifman graph for $\langle T, \Phi \rangle$ (Example 4) with a possible tree decomposition.

of $\phi$, that is, the length of the longest path from root to any leaf in the syntax tree. team-size is the cardinality of the team $T$, and formula-size is $|\phi|$. For a dependence atom $=(\boldsymbol{x};\boldsymbol{y})$ and inclusion atom $\boldsymbol{x} \subseteq \boldsymbol{y}$, the arity is defined as $|\boldsymbol{x}|$ (recall that $|\boldsymbol{x}| = |\boldsymbol{y}|$ for an inclusion atom), whereas, for an independence atom $\boldsymbol{x} \perp_{\boldsymbol{z}} \boldsymbol{y}$, it is the number of distinct variables appearing in $\boldsymbol{x} \perp_{\boldsymbol{z}} \boldsymbol{y}$. Finally, arity denotes the maximum arity of any $\mathcal{L}$-atom in $\phi$. Regarding treewidth, recall that for MC, we also include the assignment-variable relation in the graph representation. This yields two graphs: $G_\phi$ for $\phi$, and $G_{T,\phi}$ for $\langle T, \phi \rangle$. Consequently, there are two treewidth notions. formula-tw is the treewidth of $G_\phi$ and formula-team-tw is the treewidth of $G_{T,\phi}$. The name emphasises whether the team is also part of the graph. As we pointed out formula-team-tw and team-size are both only relevant for MC because an instance of SAT does not contain a team.

Given an instance $\langle T, \phi \rangle$ and a parameterisation $\kappa$, then $\kappa(T, \phi)$ denotes the parameter value of $\langle T, \phi \rangle$. The following relationship between several of the aforementioned parameters was proven for $\mathcal{PDL}$. It is easy to observe that the lemma also applies to $\mathcal{PINC}$ and $\mathcal{PIND}$.

**Lemma 5 ([22]).** *Let $\mathcal{L} \in \{\mathcal{PDL}, \mathcal{PINC}, \mathcal{PIND}\}$, $\phi$ an $\mathcal{L}$-formula and $T$ be a team. Then, team-size$(T, \phi) \leq 2^{\#\mathsf{variables}(T,\phi)}$, team-size$(T, \phi) \leq 2^{\mathsf{formula\text{-}size}(T,\phi)}$, and formula-size$(T, \phi) \leq 2^{2 \cdot \mathsf{formula\text{-}depth}(T,\phi)}$.*

Moreover, recall that we use the same graph representation for $\mathcal{PDL}, \mathcal{PINC}$ and $\mathcal{PIND}$. As a consequence, the following result also applies.

**Corollary 6 ([22]).** *Let $\mathcal{L} \in \{\mathcal{PDL}, \mathcal{PINC}, \mathcal{PIND}\}$, $\phi$ an $\mathcal{L}$-formula and $T$ be a team. Then* formula-team-tw$(T, \phi)$ *bounds team-size$(T, \phi)$.*

## 4 Complexity of inclusion and independence logic

We start with general complexity results that hold for any team based logic whose atoms are **P**-checkable. An atom $\alpha$ is **P**-checkable if given a team $T$, $T \models \alpha$ can be checked in polynomial time. It is immediate that each atom considered in this paper is **P**-checkable.

**Theorem 7.** *Let $\mathcal{L}$ be a team based logic such that $\mathcal{L}$-atoms are **P**-checkable, then* MC *for $\mathcal{L}$ when parameterized by team-size is* **FPT**.

*Proof.* We claim that the bottom up (brute force) algorithm for the model checking of $\mathcal{PDL}$ [22, Thm. 17] works for any team based logic $\mathcal{L}$ such that $\mathcal{L}$-atoms are **P**-checkable. The algorithm begins by checking the satisfaction of atoms against each subteam. This can be achieved in **FPT**-time since teamsize and consequently the number of subteams is bounded. Moreover, by taking the union of subteams for split-junction and keeping the same team for conjunction the algorithm can find subteams for each subformula in **FPT**-time. Lastly, it checks that the team $T$ is indeed a satisfying team for the formula $\phi$. For any team based logic $\mathcal{L}$, the **FPT** runtime is guaranteed if $\mathcal{L}$-atoms are **P**-checkable. Finally, the proof works for both strict and lax semantics. □

The following corollary to Theorem 7 is derived using Lemma 5 and Proposition 3.

**Corollary 8.** *Let $\mathcal{L}$ be a team based logic such that $\mathcal{L}$-atoms are* **P***-checkable, then* MC *for $\mathcal{L}$ when parameterized by $k$ is* **FPT***, if $k \in \{\mathsf{formula\text{-}team\text{-}tw}, \mathsf{formula\text{-}depth}, \#\mathsf{variables}, \mathsf{formula\text{-}size}\}$.*

The following theorem states results for satisfiability.

**Theorem 9.** *Let $\mathcal{L}$ be a team based logic s.t. $\mathcal{L}$-atoms are* **P***-checkable, then* SAT *for $\mathcal{L}$ when parameterized by $k$ is* **FPT***, if $k \in \{\mathsf{formula\text{-}depth}, \#\mathsf{variables}\}$.*

*Proof.* Notice first that the case for formula-size is trivial because any problem parameterized by input size is **FPT**. Moving on, bounding formula-depth also bounds formula-size, this yields **FPT**-membership for formula-depth in conjunction with Prop. 3. Finally, for #variables, one can enumerate all of the $2^{2^{\#\mathsf{variables}}}$-many teams in **FPT**-time and determine whether any of these satisfies the input formula. The last step requires that the model checking parameterized by team-size is **FPT**, which is true due to Theorem 7. This completes the proof. □

Our main technical contributions are the following two theorems which establish that the satisfiability problem of $\mathcal{PINC}$ parameterized by arity is **paraNP**-complete, and that SAT of $\mathcal{PINC}$ without disjunctions is tractable. We start with the former. The hardness follows from the **NP**-completeness of $\mathcal{PL}$. For membership, we give a non-deterministic algorithm $\mathbb{A}$ solving SAT.

**Theorem 10.** *There is a non-deterministic algorithm $\mathbb{A}$ that, given a $\mathcal{PINC}$-formula $\phi$ with arity $k$, runs in $O(2^k \cdot p(|\phi|))$-time and outputs a non-empty team $T$ such that $T \models \phi$ if and only if $\phi$ is satisfiable.*

*Proof.* We present the proof for lax semantics first, towards the end we describe some modifications that solve the case for strict semantics. Given an input $\mathcal{PINC}$-formula $\phi$, the algorithm $\mathbb{A}$ operates on the syntax tree of $\phi$ and constructs a sequence of teams $f_i(\psi)$ for each $\psi \in \mathrm{SF}(\phi)$ as follows. We let $f_0(\psi) := \emptyset$ for each $\psi \in \mathrm{SF}(\phi)$. Then, $\mathbb{A}$ begins by non-deterministically selecting a singleton team $f_1(\phi)$ for $\phi$. For $i \geq 1$, $\mathbb{A}$ implements the following steps recursively.

For odd $i \in \mathbb{N}$, $f_i(\psi)$ is defined in a *top-down* fashion as follows.

1. $f_i(\phi) := f_{i-1}(\phi)$ for $i \geq 3$.
2. If $\psi = \psi_0 \wedge \psi_1$, let $f_i(\psi_0) := f_i(\psi)$ and $f_i(\psi_1) := f_i(\psi)$.
3. If $\psi = \psi_0 \vee \psi_1$, then non-deterministically select two teams $P_0, P_1$ such that $P_0 \cup P_1 = f_i(\psi) \setminus f_{i-1}(\psi)$ and set $f_i(\psi_j) := f_{i-1}(\psi_j) \cup P_j$ for $j = 0, 1$.

For even $i$, $f_i(\psi)$ is defined in a *bottom-up* fashion as follows.

4. If $\psi \in \mathrm{SF}(\phi)$ is an atomic literal, then immediately reject if $f_{i-1}(\psi) \not\models \psi$ and set $f_i(\psi) := f_{i-1}(\psi)$ otherwise. If $\psi \in \mathrm{SF}(\phi)$ is an inclusion atom, then construct $f_i(\psi) \supseteq f_{i-1}(\psi)$ such that $f_i(\psi) \models \psi$. For $\psi := \boldsymbol{x} \subseteq \boldsymbol{y}$, this is done by (I) adding partial assignments $t(\boldsymbol{y}) := s(\boldsymbol{x})$ whenever an assignment $s$ is a cause for the *failure* of $\psi$, and (II) non-deterministically selecting extensions of these assignments to the other variables.
5. If $\psi = \psi_0 \wedge \psi_1$, or $\psi = \psi_0 \vee \psi_1$ let $f_i(\psi) := f_i(\psi_0) \cup f_i(\psi_1)$.

$\mathbb{A}$ terminates by accepting when a fixed point is reached. That is, we obtain $j \in \mathbb{N}$ such that $f_i(\psi) = f_{i+1}(\psi)$ for each $\psi \in \mathrm{SF}(\phi)$ when $i \geq j$. Moreover, $\mathbb{A}$ rejects if Step 4 triggers a rejection. Notice that the only step when new assignments are added is at the atomic level. Whereas the split in Step 3 concerns those assignments which arise from other subformulas through union in Step 5. We first prove the following claim regarding the overall runtime for $\mathbb{A}$.

**Claim I.** $\mathbb{A}$ runs in at most $O(2^k \cdot p(|\phi|))$ steps for some polynomial $p$, where $k$ is the arity of $\phi$. That is, a fixed point or rejection is reached in this time.

**Proof of Claim.** In each iteration $i$, either $\mathbb{A}$ rejects, or keeps adding new assignments. Furthermore, new assignments are added only in the cases for inclusion atoms. As a result, if $\mathbb{A}$ has not yet reached a fixed point the reason is that some inclusion atom has generated new assignments. Since we take union of subteams in the bottom-up step, the following top-down iteration in Steps 2 and 3 may also add assignments in a subteam. That is, the subteams from each $\psi \in \mathrm{SF}(\phi)$ are propagated to other subformulas during each iteration. Now, each inclusion atom of arity $l \leq k$ can generate at most $2^l$ new assignments due to Step 4 in the algorithm. Let $n$ denote the number of inclusion atoms in $\phi$ and $k$ be their maximum arity. Then $\mathbb{A}$ iterates at most $2^k \cdot cn$ times, where $c$ is some constant due to the propagation of teams to other subformulas. This implies that, if no rejection has occured, there is some $j \leq 2^k \cdot cn$ such that $f_i(\psi) = f_j(\psi)$ for each subformula $\psi \in \mathrm{SF}(\phi)$ and $i \geq j$. We denote this fixed point by $f_\infty(\psi)$ for each $\psi \in \mathrm{SF}(\phi)$.

Now, we analyze the time it takes to compute each iteration. For odd $i \geq 1$, Steps 1 and 2 set the same team for each subformula and therefore take linear time. Notice that the size of team in each iteration is bounded by $2^k \cdot n$. This holds because new assignments are added only in the case of inclusion atoms and $\mathbb{A}$ starts with a singleton team. Consequently, Step 3 non-deterministically splits the teams of size $2^k \cdot n$ in each iteration $i$ for odd $i \geq 1$. Moreover, Step 4 for even $i$ requires (1) polynomial time in $|f_i(\psi)|$, if $\psi$ is an atomic literal, and (2) non-deterministic polynomial time in $2^l \cdot |f_i(\psi)|$ if $\psi$ is an inclusion atom of arity $l \leq k$. Finally, the union in Step 5 again requires linear time. This implies

that each iteration takes at most a runtime of $2^k \cdot p(|\phi|)$ for some polynomial $p$. This completes the proof of Claim 1.

We now prove that $\mathbb{A}$ accepts the input formula $\phi$ if and only if $\phi$ is satisfiable. Suppose that $\mathbb{A}$ accepts and let $f_\infty(\phi)$ denote the fixed point. We first prove by induction that $f_\infty(\psi) \models \psi$ for each subformula $\psi$ of $\phi$. Notice that there is some $i$ such that $f_\infty(\psi) = f_i(\psi)$. The case for atomic subformulas is clear due to the Step 4 of $\mathbb{A}$. For conjunction, observe that the team remains the same for each conjunct. That is, when $\psi = \psi_0 \wedge \psi_1$ and the claim holds for $f_\infty(\psi_i)$ and $\psi_i$, then $f_\infty(\psi) \models \psi_0 \wedge \psi_1$ is true. For disjunction, if $\psi = \psi_0 \vee \psi_1$ and $f_\infty(\psi_i)$ are such that $f_\infty(\psi_i) \models \psi_i$ for $i = 0, 1$, then we have that $f_\infty(\psi) \models \psi$ where $f_\infty(\psi) = f_\infty(\psi_0) \cup f_\infty(\psi_1)$. In particular $f_\infty(\phi) \models \phi$ and the correctness of our algorithm follows.

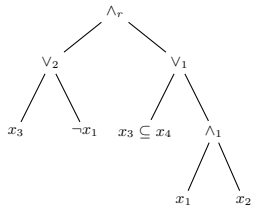
For the other direction, suppose $\phi$ is satisfiable and $T$ is a witnessing team. Then there exists a labelling function for $T$ and $\phi$, given as follows.

I. The label for $\phi$ is $T$.
II. For every subformula $\psi = \psi_0 \oplus \psi_1$ with subteam label $P \subseteq T$, the subteam label for $\psi_i$ is $P_i$ $(i = 0, 1)$ such that we have $P_0 = P_1 = P$, if $\oplus = \wedge$, and $P_0 \cup P_1 = P$ if $\oplus = \vee$,
III. $P_\psi \models \psi$ for every $\psi \in \mathrm{SF}(\phi)$ with label $P_\psi$.

Then we prove that there exists an accepting path when the non-deterministic algorithm $\mathbb{A}$ operates on $\phi$. We claim that when initiated on a subteam $\{s\} \subseteq T$, $\mathbb{A}$ constructs a fixed point $f_\infty(\phi)$ and halts by accepting $\phi$. Recall that $\mathbb{A}$ propagates teams back and forth until a fixed point is reached. Moreover, the new assignments are added only at the atomic level. Let $\alpha := \boldsymbol{x} \subseteq \boldsymbol{y}$ be an inclusion atom such that $f_i(\alpha) \neq \emptyset$ for odd $i$, then $\mathbb{A}$ constructs a subteam $f_{i+1}(\alpha) \supseteq f_i(\alpha)$ (on a non-deterministic branch) containing assignments $t$ from $P_\alpha$ such that $f_{i+1}(\alpha) \models \alpha$. Since, there are at most $2^{|\boldsymbol{y}|}$-many different assignments for $\boldsymbol{y}$, we know that Step 4 applies to $\alpha$ at most $2^{|\boldsymbol{y}|}$ times. That is, once all the different assignments for $\boldsymbol{y}$ have been checked in some iteration $i$: Step 4 does not add any further assignments to $f_{i'}(\alpha)$ for $i' \geq i + 1$. Finally, since there is a non-empty team $T$ such that $T \models \phi$, this implies that $\mathbb{A}$ does not reject $\phi$ in any iteration (because there is a choice for $\mathbb{A}$ to consider subteams guaranteed by the labelling function). Consequently, $\mathbb{A}$ accepts by constructing a fixed point in at most $O(2^k \cdot p(|\phi|))$-steps (follows from Claim I). This completes the proof and establishes the correctness.

A minor variation in the algorithm $\mathbb{A}$ solves SAT for the strict semantics. When moving downwards, $\mathbb{A}$ needs to ensure that an assignment goes to only one side of the split. Moreover, since the subteams are selected non-deterministically for atomic subformulas, (in the bottom-up iteration) only subteams which can split according to the strict semantics are considered. □

*Example 11.* We include an example to explain how $\mathbb{A}$ from the proof of Theorem 10 operates. Figure 3 depicts the steps of $\mathbb{A}$ on a formula $\phi$. An assignment over $\{x_1, \ldots, x_4\}$ is seen as a tuple of length four. It is easy to observe that the third iteration already yields a fixed point and that $f_3(\psi) = f_4(\psi)$ for each $\psi \in \mathrm{SF}(\phi)$. In this example, the initial guess made by $\mathbb{A}$ is the team $\{0110\}$.




| $\psi$ | $f_1(\psi)$ | $f_2(\psi)$ | $f_3(\psi)$ |
|---|---|---|---|
| $\phi$ | 0110 | 0110,1111 | 0110,1111 |
| $\vee_2$ | 0110 | 0110 | 0110,1111 |
| $\vee_1$ | 0110 | 0110,1111 | 0110,1111 |
| $x_3 \subseteq x_4$ | 0110 | 0110,1111 | 0110,1111 |
| $x_1 \wedge x_2$ | | | 1111 |
| $x_3$ | 0110 | 0110 | 0110,1111 |
| $\neg x_1$ | 0110 | 0110 | 0110 |

**Fig. 3.** The table (Right) indicates subteams for each $\psi \in \mathrm{SF}(\phi)$ (Left). The teams $f_1(\psi)$ and $f_3(\psi)$ are propagated top-down whereas $f_2(\psi)$ is propagated bottom-up. For brevity we omit subformulas $x_1$ and $x_2$ of $x_1 \wedge x_2$.

The following corollaries follow immediately from the proof of Theorem 10.

**Corollary 12.** *Given a $\mathcal{PINC}$-formula $\phi$ with arity $k$, then $\phi$ is satisfiable if and only if there is a team $T$ of size at most $O(2^k \cdot p(|\phi|))$ such that $T \models \phi$.*

*Proof.* Simulate the algorithm $\mathbb{A}$ from the proof of Theorem 10. Since $\phi$ is satisfiable, $\mathbb{A}$ halts in at most $O(2^k \cdot p(|\phi|))$-steps and thereby yields a team (namely, $f_\infty(\phi)$) of the given size. □

**Corollary 13.** SAT *for $\mathcal{PINC}$, when parameterized by* formula-tw *of the input formula is in* **paraNP**.

*Proof.* Recall the Graph structure where we allow edges between variables within an inclusion atom. This implies that for each inclusion atom $\alpha$, there is a bag in the tree decomposition that contains all variables of $\alpha$. As a consequence, a formula $\phi$ with treewidth $k$ has inclusion atoms of arity at most $k$. Consequently, SAT parameterized by treewidth of the input formula can be solved using the **paraNP**-time algorithm from the proof of Theorem 10. □

Regarding the parameter #splits, the precise parameterized complexity is still open for now. However, we prove that if there is no split in the formula, then SAT can be solved in polynomial time. This case is interesting in its own right because it gives rise to the so-called Poor Man's $\mathcal{PINC}$, similar to the case of Poor Man's $\mathcal{PDL}$ [6,21,23]. The model checking for this fragment is in **P**; this follows from the fact that MC for $\mathcal{PINC}$ with lax semantics is in **P**. In the following, we prove that SAT for Poor Man's $\mathcal{PINC}$ is also in **P**.

**Theorem 14.** *There is a deterministic algorithm $\mathbb{B}$ that given a $\mathcal{PINC}$-formula $\phi$ with no splits runs in* **P***-time and accepts if and only if $\phi$ is satisfiable.*

*Proof.* We give a recursive labelling procedure ($\mathbb{B}$) that runs in polynomial time and accepts if and only if $\phi$ is satisfiable. The labelling consists of assigning a value $c \in \{0, 1\}$ to each variable $x$.

1. Begin by labelling all $\mathcal{PL}$-literals in $\phi$ by the value that satisfies them, namely $x = 1$ for $x$ and $x = 0$ for $\neg x$.

2. For each inclusion atom $\boldsymbol{p} \subseteq \boldsymbol{q}$ and a labelled variable $q_i \in \boldsymbol{q}$, label the variable $p_i \in \boldsymbol{p}$ with same value $c$ as for $q_i$. Where $p_i$ appears in $\boldsymbol{p}$ at the same position, as $q_i$ in $\boldsymbol{q}$.
3. Propagate the label for $p_i$ from the previous step. That is, consider $p_i$ as a labelled variable and repeat Step 2 for as long as possible.
4. If some variable $x$ is labelled with two opposite values, then reject. Otherwise, accept.

The fact that $\mathbb{B}$ works in polynomial time is clear because each variable is labelled at most once. If a variable is labelled to two different values, then it gives a contradiction and the procedure stops.

For the correctness, notice first that if $\mathbb{B}$ accepts then we have a partition of $\mathrm{Var}(\phi)$ into a set $X$ of labelled variables and a set $Y = \mathrm{Var}(\phi)\backslash X$. When $\mathbb{B}$ stops, due to step 3, $\phi$ does not contain an inclusion atom $\boldsymbol{p} \subseteq \boldsymbol{q}$ such that $q_i \in \boldsymbol{q}$ and $p_i \in \boldsymbol{p}$ for some $q_i \in X, p_i \in Y$, where $p_i$ appears in $\boldsymbol{p}$ at the same position as $q_i$ in $\boldsymbol{q}$. Let $T = \{s \in 2^{\mathrm{Var}(\phi)} \mid x \text{ is labelled with } s(x), \text{ for each } x \in X\}$. Since $\mathbb{B}$ accepts, each variable $x \in X$ has exactly one label and therefore assignments in $T$ are well-defined. Moreover $T$ includes all possible assignments over $Y$. One can easily observe that $T \models \phi$. $T$ satisfies each literal because each $s \in T$ satisfies it. Let $\boldsymbol{p} \subseteq \boldsymbol{q}$ be an inclusion atom and $s \in T$ be an assignment. We know that for each $x \in \boldsymbol{q}$ that is fixed by $s$, the corresponding variable $y \in \boldsymbol{p}$ is also fixed, whereas, $T$ contains every possible value for variables in $\boldsymbol{q}$ which are not fixed. This makes the inclusion atom true.

To prove the other direction, suppose that $\mathbb{B}$ rejects. Then there are three cases under which a variable contains contradictory labels. Either both labels of the variable are caused by a literal (Case 1), or inclusion atoms are involved in one (Case 2), or both (Case 3) labels. In other words, either $\phi$ contains $x \wedge \neg x$ as a subformula, or it contains $x \wedge \neg y$ and there is a sequence of inclusion atoms, such that keeping $x = 1$ and $y = 0$ contradicts some inclusion atoms in $\phi$ (see Figure 4).

**Case 1** Both labels of a variable $x$ are caused by a literal. In this case, $x$ takes two labels because $\phi$ contains $x \wedge \neg x$. The proof is trivial since $\phi$ is unsatisfiable.

**Case 2** One label of a variable $y$ is caused by a literal ($\neg y$ or $y$) and the other by inclusion atoms. Then, there are inclusion atoms $\boldsymbol{p}_j \subseteq \boldsymbol{q}_j$ and variables $z_j$ for $j \leq n$ such that: $z_0 = x$, $z_n = y$, and $z_j$ and $z_{j+1}$ occur in the same position in $\boldsymbol{q}_j$ and $\boldsymbol{p}_j$, respectively, for $0 \leq j < n$. This implies that $\phi$ is not satisfiable since for any team $T$ such that $T \models x \wedge \neg y$, $T$ does not satisfy the subformula $\bigwedge_j \boldsymbol{p}_j \subseteq \boldsymbol{q}_j$ of $\phi$. A similar reasoning applies if $\phi$ contains $\neg x \wedge y$ instead.

**Case 3** Both labels of a variable $v$ are caused by inclusion atoms. Then, there are two collections of inclusion atoms $\boldsymbol{p}_j \subseteq \boldsymbol{q}_j$ for $j \leq n$, and $\boldsymbol{r}_k \subseteq \boldsymbol{s}_k$ for $k \leq m$. Moreover, there are two sequences of variables $z^x_j$ for $j \leq n$ and $z^y_k$ for $k \leq m$, and a variable $v$ such that, $z^x_0 = x$, $z^y_0 = y$, $z^x_n = v = z^y_m$, and

1. for each $j \leq n$, $z^x_j$ appears in $\boldsymbol{q}_j$ at the same position, as $z^x_{j+1}$ in $\boldsymbol{p}_j$,
2. for each $k \leq m$, $z^y_k$ appears in $\boldsymbol{s}_k$ at the same position, as $z^y_{k+1}$ in $\boldsymbol{r}_k$.

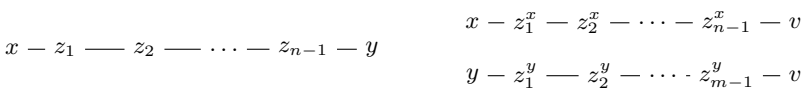



**Fig. 4.** Intuitive explanation of two cases in the proof. (Left) $x$ and $\neg y$ propagate a conflicting value to eachother. (Right) $x$ and $\neg y$ propagate conflicting values to $v$.

| $\mathrm{Var}(\phi)$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| Labels for $\mathrm{Var}(\phi)$ | 1 | 1 | | 0 | |
| Propagation due to $x_5 \subseteq x_4$ | 1 | 1 | | 0 | 0 |
| Propagation due to $x_1x_3 \subseteq x_5x_2$ | 1/0 | 1 | 1 | 0 | 0 |

**Fig. 5.** Labels for literals and their propagation to inclusion atoms (See Example 15).

This again implies that $\phi$ is not satisfiable since for any $T$ such that $T \models x \wedge \neg y$, it does not satisfy the subformula $\bigwedge_j \boldsymbol{p}_j \subseteq \boldsymbol{q}_j \wedge \bigwedge_k \boldsymbol{r}_k \subseteq \boldsymbol{s}_k$ of $\phi$.

Consequently, the correctness follows. This completes the proof. □

*Example 15.* We include an example to highlights how $\mathbb{B}$ operates. Let $\phi := (x_1 \wedge x_2 \wedge \neg x_4) \wedge (x_1x_3 \subseteq x_5x_2) \wedge (x_5 \subseteq x_4)$. The table in Figure 5 illustrates the steps of $\mathbb{B}$ on $\phi$. Clearly, $\mathbb{B}$ rejects $\phi$ since the variable $x_1$ has conflicting labels.

The **FPT** cases for SAT of $\mathcal{PINC}$ follow from Theorem 9. Regarding MC, recall that we consider strict semantics alone. The results of Theorem 17 are obtained from the reduction for proving **NP**-hardness of $\mathrm{MC}_s$ for $\mathcal{PINC}$ [14]. Here we confirm that their reduction is indeed an *fpt*-reduction with respect to considered parameters. The following lemma is essential for proving Theorem 17 and we include it for self containment.

**Lemma 16 ([14]).** MC *for* $\mathcal{PINC}$ *under strict semantics is* **NP***-hard.*

*Proof Idea.* The hardness is achieved through a reduction from the set splitting problem to the model checking problem for $\mathcal{PINC}$ with strict semantics. An instance of set splitting problem consists of a family $\mathcal{F}$ of subsets of a finite set $S$. The problem asks if there are $S_1, S_2 \subseteq S$ such that $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ and for each $A \in \mathcal{F}$ there exists $a_1, a_2 \in \mathcal{A}$ such that $a_1 \in S_1, a_2 \in S_2$. Let $\mathcal{F} = \{B_1, \ldots, B_n\}$ and $\bigcup \mathcal{F} = S = \{a_1, \ldots, a_k\}$. Let $p_i$ and $q_j$ denote fresh variables for each $a_i \in S$ and $B_j \in \mathcal{F}$. Moreover, let $V_{\mathcal{F}} = \{p_1, \ldots, p_k, q_1, \ldots, q_n, p_\top, p_c, p_d\}$. Then define $T_{\mathcal{F}} = \{s_1, \ldots, s_k, s_c, s_d\}$, where each assignment $s_i$ is defined as follows:

$$s_i(p) := \begin{cases} 1, \text{ if } p = p_i \text{ or } p = p_\top, \\ 1, \text{ if } p = q_j \text{ and } a_i \in B_j \text{ for some } j, \\ 0, \text{ otherwise.} \end{cases}$$

That is, $T_{\mathcal{F}}$ includes an assignment $s_i$ for each $a_i \in S$. The reduction also yields the following $\mathcal{PINC}$-formula.

$$\phi_{\mathcal{F}} := (\neg p_c \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i) \vee (\neg p_d \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i)$$

Clearly, the split of $T_{\mathcal{F}}$ into $T_1, T_2$ ensures the split of $S$ into $S_1$ and $S_2$ and vice versa. Whereas, $s_c$ and $s_d$ ensure that none of the split is empty. □

**Theorem 17.** $\mathrm{MC}_s$ *for* $\mathcal{PINC}$ *when parameterized by* $k$ *is* **paraNP***-complete if* $k \in \{\#\mathsf{splits}, \mathsf{arity}, \mathsf{formula\text{-}tw}\}$. *Whereas, it is* **FPT** *in other cases.*

*Proof.* Consider the $\mathcal{PINC}$-formula $\phi_{\mathcal{F}}$ from Lemma 16, which includes only one split-junction and the inclusion atoms have arity one. This gives the desired **paraNP**-hardness for $\mathrm{MC}_s$ when parameterized by #splits and arity.

The proof for formula-tw is more involved and we prove the following claim.

*Claim.* $\phi_{\mathcal{F}}$ has fixed formula-tw. That is, the treewidth of $\phi_{\mathcal{F}}$ is independent of the input instance $\mathcal{F}$ of the set-spliting problem. Moreover $\mathsf{formula\text{-}tw}(\phi_{\mathcal{F}}) \leq 4$.

*Proof of Claim.* The $\mathcal{PINC}$-formula $\phi_{\mathcal{F}}$ is related to an input instance $\mathcal{F}$ of the set splitting problem only through its input size, which is $n$. Therefore the formula structure remains unchanged when we vary an input instance, only the size of two big conjunctions vary. To prove the claim, we give a tree decomposition for the formula with $\mathsf{formula\text{-}tw}(\phi_{\mathcal{F}}) = 4$. Since the treewidth is minimum over all tree decompositions, this proves the claim. We rewrite the formula as below.

$$\phi_{\mathcal{F}} := (\neg p_c \wedge_l \bigwedge_{i \leq n} p_\top \subseteq^l_i q_i) \vee (\neg p_d \wedge_r \bigwedge_{i \leq n} p_\top \subseteq^r_i q_i)$$

That is, each subformula is renamed so that it is easy to identify as to which side of the split it appears (e.g., $p_\top \subseteq^l_i q_i$ denotes the $i$th inclusion atom in the big conjunction on the left, denoted as $I^l_i$ in the graph). The graphical representation of $\phi_{\mathcal{F}}$ with $V = \mathrm{SF}(\phi_{\mathcal{F}}) \cup \mathrm{Var}(\phi_{\mathcal{F}})$, as well as, a tree decomposition, is given in Figure 6. Notice that there is an edge between $x$ and $y$ in the Gaifman graph if and only if either $y$ is an immediate subformula of $x$, or $y$ is a variable appearing in the inclusion atom $x$. It is easy to observe that the decomposition presented in Figure 6 is indeed a valid tree decomposition in which each node is labelled with its corresponding bag. Moreover, since the maximum bag size is 5, the treewidth of this decomposition is 4. This proves the claim. ■

The remaining **FPT**-cases for $\mathrm{MC}_s$ follow from Theorem 7 and Corollary 8. This completes the proof to our theorem. □

Recall that a dependence atom $=(\boldsymbol{x}; \boldsymbol{y})$ is equivalent with the independence atom $\boldsymbol{y} \perp_{\boldsymbol{x}} \boldsymbol{y}$. As a consequence, (in the classical setting) hardness results for $\mathcal{PDL}$ immediately translate to those for $\mathcal{PIND}$. Nevertheless, in the parameterized setting, one has to further check whether this translation 'respects' the parameter value of the two instances. This concerns the parameter arity and formula-tw. This is due to the reason that, a dependence atom $=(\boldsymbol{x}; \boldsymbol{y})$ has arity $|\boldsymbol{x}|$, whereas, the equivalent independence atom $\boldsymbol{y} \perp_{\boldsymbol{x}} \boldsymbol{y}$ has arity $|\boldsymbol{x} \cup \boldsymbol{y}|$.

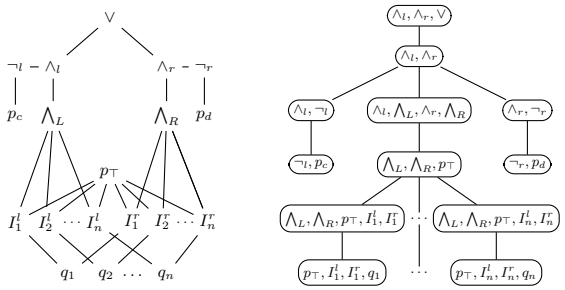



**Fig. 6.** The Gaifman graph (Left) and a tree decomposition (Right) for $\phi_{\mathcal{F}}$. Note that we abbreviated subformulas in the inner vertices of the Gaifman graph for presentation reasons. Also, edges between $p_\top$ and variables $q_i$ are omitted for better presentation, but those are covered in the decomposition on the right.

**Theorem 18.** MC *for* $\mathcal{PIND}$*, when parameterized by* $k$ *is* **paraNP***-complete if* $k \in \{\#\mathsf{splits}, \mathsf{arity}, \mathsf{formula\text{-}tw}\}$*. Whereas, it is* **FPT** *in other cases.*

*Proof.* Notice that MC for $\mathcal{PDL}$ when parameterized by $k \in \{\mathsf{arity}, \#\mathsf{splits}, \mathsf{formula\text{-}tw}\}$ is also **paraNP**-complete. We argue that in reductions for $\mathcal{PDL}$, replacing dependence atoms by the equivalent independence atoms yield *fpt*-reduction for the above mentioned cases. Moreover, this holds for both strict and lax semantics.

For formula-tw and arity, when proving **paraNP**-hardness of $\mathcal{PDL}$, the resulting formula has treewidth of one [22, Cor. 16] and the arity is zero [22, Thm. 15]. Moreover, only dependence atoms of the form $=(;p)$ where $p$ is a propositional variable, are used and the syntax structure of the $\mathcal{PDL}$-formula is already a tree. Consequently, replacing $=(;p)$ with $p\perp_\emptyset p$ implies that only independence atoms of arity 1 are used. Notice also that replacing dependence atoms by independence atoms does not increase the treewidth of the input formula. This is because when translating dependence atoms into independence atoms, no new variables are introduced. As a result, the reduction also preserves the treewidth. This proves the claim as 1-slice regarding both parameters arity and formula-tw, is **NP**-hard.

Regarding the #splits, the claim follows due to Mahmood and Meier [22, Thm. 18] because the reduction from the colouring problem uses only 2 splits.

Finally, the **FPT** cases follow from Theorem 7 and Corollary 8. □

**Theorem 19.** SAT *for* $\mathcal{PIND}$*, parameterized by* arity *is* **paraNP***-complete. Whereas, it is* **FPT** *in other cases.*

*Proof.* Recall that $\mathcal{PL}$ is a fragment of $\mathcal{PIND}$. This immediately gives **paraNP**-hardness when parameterized by arity, because SAT for $\mathcal{PL}$ is **NP**-complete. The **paraNP**-membership is clear since SAT for $\mathcal{PIND}$ is also **NP**-complete [12, Thm. 1]. The **FPT** cases for $k \in \{\mathsf{formula\text{-}depth}, \#\mathsf{variables}\}$ follow because of Theorem 9 . The cases for #splits and formula-tw follow due to a similar reasoning

as in $\mathcal{PDL}$ [22] because it is enough to find a singleton satisfying team [13, Lemma 4.2]. This completes the proof. □

## 5 Concluding Remarks

We presented a parameterized complexity analysis for $\mathcal{PINC}$ and $\mathcal{PIND}$. The problems we considered were satisfiability and model checking. Interestingly, the parameterized complexity results for $\mathcal{PIND}$ coincide with that of $\mathcal{PDL}$ [22] in each case. Moreover, the complexity of model checking under a given parameter remains the same for all three logics. We proved that for a team based logic $\mathcal{L}$ such that $\mathcal{L}$-atoms can be evaluated in **P**-time, MC for $\mathcal{L}$ when parameterized by team-size is always **FPT**.

It is interesting to notice that for $\mathcal{PDL}$ and $\mathcal{PIND}$, SAT is easier than MC when parameterized by formula-tw. This is best explained by the fact that $\mathcal{PDL}$ is downwards closed and a formula is satisfiable iff some singleton team satisfies it. Moreover, $\mathcal{PIND}$ also satisfies this 'satisfiable under singleton team' property. The parameters team-size and formula-team-tw are not meaningful for SAT due to the reason that we do not impose a size restriction for the satisfying team in SAT. Furthermore, arity is quite interesting because SAT for all three logics is **paraNP**-complete. This implies that while the fixed arity does not lower the complexity of SAT in $\mathcal{PDL}$ and $\mathcal{PIND}$, it does lower it from **EXP**-completeness to **NP**-completeness for $\mathcal{PINC}$. As a byproduct, we obtain that the complexity of satisfiability for the fixed arity fragment of $\mathcal{PINC}$ is **NP**-complete. Thereby, we answer an open question posed by Hella and Stumpf [15, P.13]. The **paraNP**-membership of SAT when parameterized by arity implies that one can encode the problem into classical satisfiability and employ a SAT-solver to solve satisfiability for the fixed arity fragment of $\mathcal{PINC}$. We leave as a future work the suitable SAT-encoding for $\mathcal{PINC}$ that runs in **FPT**-time and enables one to use SAT-solvers. Further future work involves finding the precise complexity of SAT for $\mathcal{PINC}$ when parameterized by #splits and formula-tw.

### Acknowledgement

This work was supported by the European Union's Horizon Europe research and innovation programme within project ENEXA (101070305) and by the German Research Foundation (DFG), project VI 1045/1-1.

## References

1. Corander, J., Hyttinen, A., Kontinen, J., Pensar, J., Väänänen, J.: A logical approach to context-specific independence. Ann. Pure Appl. Logic **170**(9), 975–992 (2019). https://doi.org/10.1016/j.apal.2019.04.004, https://doi.org/10.1016/j.apal.2019.04.004
2. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer (2013). https://doi.org/10.1007/978-1-4471-5559-1, https://doi.org/10.1007/978-1-4471-5559-1

3. Durand, A., Hannula, M., Kontinen, J., Meier, A., Virtema, J.: Approximation and dependence via multiteam semantics. Ann. Math. Artif. Intell. **83**(3-4), 297–320 (2018). https://doi.org/10.1007/s10472-017-9568-4, https://doi.org/10.1007/s10472-017-9568-4
4. Durand, A., Hannula, M., Kontinen, J., Meier, A., Virtema, J.: Probabilistic team semantics. In: FoIKS (2018). https://doi.org/10.1007/978-3-319-90050-6_11
5. Ebbing, J., Hella, L., Meier, A., Müller, J., Virtema, J., Vollmer, H.: Extended modal dependence logic. In: WoLLIC (2013). https://doi.org/10.1007/978-3-642-39992-3_13, http://dx.doi.org/10.1007/978-3-642-39992-3_13
6. Ebbing, J., Lohmann, P.: Complexity of model checking for modal dependence logic. In: SOFSEM (2012). https://doi.org/10.1007/978-3-642-27660-6_19, https://doi.org/10.1007/978-3-642-27660-6_19
7. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series, Springer (2006)
8. Galliani, P.: Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. Ann. Pure Appl. Logic **163**(1), 68–84 (2012). https://doi.org/10.1016/j.apal.2011.08.005, https://doi.org/10.1016/j.apal.2011.08.005
9. Hannula, M., Kontinen, J., Van den Bussche, J., Virtema, J.: Descriptive complexity of real computation and probabilistic independence logic. In: LICS '20. pp. 550–563. ACM (2020). https://doi.org/10.1145/3373718.3394773, https://doi.org/10.1145/3373718.3394773
10. Hannula, M., Kontinen, J., Lück, M., Virtema, J.: On quantified propositional logics and the exponential time hierarchy. In: GandALF. EPTCS, vol. 226, pp. 198–212 (2016)
11. Hannula, M., Kontinen, J., Virtema, J.: Polyteam semantics. J. Log. Comput. **30**(8), 1541–1566 (2020). https://doi.org/10.1093/logcom/exaa048, https://doi.org/10.1093/logcom/exaa048
12. Hannula, M., Kontinen, J., Virtema, J., Vollmer, H.: Complexity of propositional independence and inclusion logic. In: MFCS 2015,. Lecture Notes in Computer Science, vol. 9234, pp. 269–280. Springer (2015). https://doi.org/10.1007/978-3-662-48057-1_21, http://dx.doi.org/10.1007/978-3-662-48057-1_21
13. Hannula, M., Kontinen, J., Virtema, J., Vollmer, H.: Complexity of propositional logics in team semantic. ACM Trans. Comput. Log. **19**(1), 2:1–2:14 (2018). https://doi.org/10.1145/3157054, https://doi.org/10.1145/3157054
14. Hella, L., Kuusisto, A., Meier, A., Virtema, J.: Model checking and validity in propositional and modal inclusion logics. J. Log. Comput. **29**(5), 605–630 (2019). https://doi.org/10.1093/logcom/exz008, https://doi.org/10.1093/logcom/exz008
15. Hella, L., Stumpf, J.: The expressive power of modal logic with inclusion atoms. In: Proceedings of the 6th GandALF. pp. 129–143 (2015)
16. Hyttinen, T., Paolini, G., Väänänen, J.: A Logic for Arguing About Probabilities in Measure Teams. Arch. Math. Logic **56**(5-6), 475–489 (2017). https://doi.org/10.1007/s00153-017-0535-x
17. Kontinen, J., Mahmood, Y., Meier, A., Vollmer, H.: Parameterized complexity of weighted team definability (2023), https://doi.org/10.48550/arXiv.2302.00541
18. Kontinen, J., Meier, A., Mahmood, Y.: A parameterized view on the complexity of dependence and independence logic. J. Log. Comput. **32**(8), 1624–1644 (2022). https://doi.org/10.1093/logcom/exac070, https://doi.org/10.1093/logcom/exac070

19. Krebs, A., Meier, A., Virtema, J.: A team based variant of CTL. In: TIME 2015. pp. 140–149 (2015). https://doi.org/10.1109/TIME.2015.11, http://dx.doi.org/10.1109/TIME.2015.11
20. Krebs, A., Meier, A., Virtema, J., Zimmermann, M.: Team Semantics for the Specification and Verification of Hyperproperties. In: MFCS 2018. vol. 117, pp. 10:1–10:16. Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.MFCS.2018.10
21. Lohmann, P., Vollmer, H.: Complexity results for modal dependence logic. Studia Logica **101**(2), 343–366 (2013). https://doi.org/10.1007/s11225-013-9483-6, https://doi.org/10.1007/s11225-013-9483-6
22. Mahmood, Y., Meier, A.: Parameterised complexity of model checking and satisfiability in propositional dependence logic. Annals of Mathematics and Artificial Intelligence (2021). https://doi.org/10.1007/s10472-021-09730-w, https://doi.org/10.1007/s10472-021-09730-w
23. Meier, A., Reinbold, C.: In: Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings. https://doi.org/10.1007/978-3-319-90050-6_17
24. Peterson, G., Reif, J., Azhar, S.: Lower bounds for multiplayer noncooperative games of incomplete information. Computers & Mathematics with Applications **41**(7), 957–992 (2001)
25. Peterson, G.L., Reif, J.H.: Multiple-person alternation. pp. 348–363. IEEE Computer Society (1979). https://doi.org/10.1109/SFCS.1979.25, https://doi.org/10.1109/SFCS.1979.25
26. Pippenger, N.: Theories of computability. Cambridge University Press (1997)
27. Shukla, A., Biere, A., Pulina, L., Seidl, M.: A survey on applications of quantified boolean formulas. In: ICTAI 2019. https://doi.org/10.1109/ICTAI.2019.00020, https://doi.org/10.1109/ICTAI.2019.00020
28. Väänänen, J.: Dependence Logic. Cambridge University Press (2007)
29. Väänänen, J., Hodges, W.: Dependence of Variables Construed as an Atomic Formula (2008)
30. Virtema, J.: Complexity of validity for propositional dependence logics. Inf. Comput. **253**, 224–236 (2017). https://doi.org/10.1016/j.ic.2016.07.008, https://doi.org/10.1016/j.ic.2016.07.008