Model checking and validity in propositional and modal inclusion logics

Model checking and validity in propositional and modal inclusion logics

LAURI HELLA, *Unit of Computing Sciences, Tampere University, 33100 Tampere, Finland.*
E-mail: lauri.hella@tuni.fi

ANTTI KUUSISTO, *Unit of Computing Sciences, Tampere University, 33100 Tampere, Finland.*
E-mail: antti.kuusisto@tuni.fi

ARNE MEIER, *Institut für Theoretische Informatik, Leibniz Universität Hannover, 30167 Hanover, Germany.*
E-mail: meier@thi.uni-hannover.de

JONNI VIRTEMA, *Department of Mathematics and Statistics, University of Helsinki, 00014 Helsinki, Finland, and Faculty of Sciences, Hasselt University, 3500 Hasselt, Belgium.*
E-mail: jonni.virtema@gmail.com

Abstract

Propositional and modal inclusion logic are formalisms that belong to the family of logics based on team semantics. This article investigates the model checking and validity problems of these logics. We identify complexity bounds for both problems, covering both lax and strict team semantics. By doing so, we come close to finalizing the programme that aims to completely classify the complexities of the basic reasoning problems for modal and propositional dependence, independence and inclusion logics.

Keywords: Inclusion logic, model checking, validity problem, complexity, team semantics

1 Introduction

Team semantics is the mathematical framework of logics of dependence and independence proposed by Väänänen [39] and inspired by Hodges [21]. In Tarskian semantics, satisfaction of formulae is defined with respect to single states of affairs; e.g. with respect to a single assignment in first-order and propositional logic, and with respect to a single point of a Kripke structure in modal logic. Team semantics is founded on the simple shift from single states of affairs to a collection of states of affairs as the principal object associated with satisfaction. In the first-order setting, logics with team semantics are very expressive. The so-called first-order *independence logic* is equi-expressive with existential second-order logic and thus captures the complexity class NP [9]. On the other hand, the so-called first-order *inclusion logic* has the same expressive power as *positive greatest fixed point logic* GFP^+ [10] and, by the Immermann-Vardi Theorem [22, 40], it captures P on

Vol. 00, No. 0,
doi:10.1093/logcom/exz008

ordered structures. In the modal setting, the expressive powers of related logics are characterized by the so-called *team k-bisimulation*. During the past decade, the research on team semantics has flourished, many logical formalisms have been defined and surprising connections to other fields have been identified. For instance, Krebs *et al.* [25] developed a team-based approach to linear temporal logic for the verification of information flow properties. In applications to database theory, a team corresponds exactly to a database table (see, e.g. the work of Hannula and Kontinen [14]). Hannula *et al.* [15] introduced a framework that extends the connection of team semantics and database theory to polyrelational databases and data exchange. The study of inclusion dependencies in database theory is a highly relevant topic. The implication problem for inclusion dependencies was axiomatized by Casanova, Fagin and Papadimitriou [3], and recently Koehler and Link [23] gave a finite axiomatization for the so-called *not null inclusion dependencies*.

The high expressivity and complexity of logics with team semantics (see the survey of Durand *et al.* [7]) have led the research community to consider weaker logics and weak fragments of logics with team semantics. From this endeavour the study of propositional and modal team semantics have risen as research topics of their own right. In contrast to their first-order counterparts, propositional and modal logics with team semantics are decidable and enjoy finite axiomatizations (for concrete axiomatizations, see e.g. [29, 33, 43]). A propositional team is a set of propositional assignments with a common finite domain of variables. In the modal context, any subset of the domain of a Kripke structure is a team. The most studied logics in the propositional and modal team semantics contexts are propositional (modal, resp.) dependence, independence and inclusion logics, and their extension with the contradictory negation. Here we concentrate on inclusion logics. *Propositional* (*modal*, resp.) *inclusion logic* extends propositional (modal, resp.) logic with inclusion atoms of the form

$$(p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n) \tag{1}$$

with the semantics that a team X satisfies the atom in (1) if every tuple of truth values that occurs in X for $(p_1, \ldots, p_n)$ also occurs as a truth value for $(q_1, \ldots, q_n)$ in X. In the extended propositional (modal, resp.) inclusion logic, the variables in (1) are replaced by formulae of propositional (modal, resp.) logic.

Due to very active research efforts, the complexity and definability landscape of propositional and modal logics with team semantics is understood rather well, see the survey of Durand *et al*. [7] and Tables 4–6 on page 23 of the current article. For basic properties of propositional and modal inclusion logics, see the work by Yang and Väänänen [43] and the work by Hella and Stumpf [20], respectively. One characteristic feature of inclusion logics is the following union closure property: if teams X and Y both satisfy an inclusion logic formula φ, then also then the union $X \cup Y$ of X and Y satisfies the formula φ. Hella and Stumpf [20] established that the expressive power of modal inclusion logic can be exactly characterized by the union closure and the closure under the so-called *team k-bisimulation* (the concept of team bisimulation is a straightforward generalization of the notion of standard bisimulation of modal logic). As a consequence, it follows that the expressive power of extended propositional inclusion logic is exactly characterized by the union closure property. Sano and Virtema [34, 35] study (global) model definability and frame definability of team-based modal logics. The authors show that, surprisingly, in both cases, extended modal inclusion logic collapses to modal logic. This is surprising since, in contrast, the so-called *extended model dependence logic* is shown to correspond to the extension of modal logic with the universal modality defined using Tarski semantics.

In this paper, we study propositional and modal inclusion logic under both *lax semantics* (i.e. the standard semantics) and *strict semantics*. The difference in strict and lax semantics can

be seen in the semantic conditions for the disjunction and the diamond (see Sections 2 and 4 for the technical details); the characterizations of expressive power, mentioned in the last paragraph, hold only for lax semantics. Many of the desired properties that lax semantics satisfies fail when strict semantics is considered. For instance, under strict semantics propositional and modal inclusion logic do not satisfy the union closure property (see Example 5). What is worse, propositional inclusion logic under strict semantics does not satisfy locality: satisfaction of an inclusion logic formula on a team X is not determined alone by the restriction of X to the set of variables that occur in the formula. These problems were already identified in the first-order setting. However, recent work on multiteam semantics (a multiteam is a multiset analogue of a team) has revealed that the problems in strict semantics are more subtle and context specific; many problems of strict semantics are solved naturally in multiteams. In particular, under the so-called strict multiteam semantics inclusion logic satisfies locality and union closure [6]. While the setting of [6] is first-order team semantics the results and ideas there can be modified and transferred also to the propositional and modal settings.

We investigate the complexity of the model checking and the validity problem for propositional and modal inclusion logic. Our aim is to completely determine the complexity landscape of propositional and modal inclusion logics. By doing this, we come close to finalizing the programme that aims to completely classify the complexities of the basic reasoning problems for modal and propositional dependence, independence and inclusion logics (see the Tables 4–6 on page 23). The complexity of the satisfiability problem of modal inclusion logic was studied by Hella *et al.* [18]. The study of the validity problem of propositional inclusion logic was initiated by Hannula *et al.* [16], where the focus was on more expressive logics in the propositional setting. The current paper directly extends the research effort initiated in these papers. It is important to note that, since the logics studied in this paper are not closed under taking negations, the standard connection between the satisfiability problem and the validity problem fails (i.e. a formula is unsatisfiable if and only if its negation is a tautology). Hannula *et al.* [16] have shown that, under lax semantics, the validity problem for propositional inclusion logic is coNP-complete. Here we obtain an analogous result for the strict semantics. Curiously, for model checking the picture looks quite different. We establish that whereas the model checking problem for propositional inclusion logic is P-complete under lax semantics, the problem becomes NP-complete for the strict variant. Intriguingly, for model checking in the modal setting, we obtain analogous results (as in the propositional setting): modal inclusion logic is P-complete under lax semantics and NP-complete under strict semantics. Nevertheless, for the validity problem, the modal variants are much more complex than the propositional ones: we establish coNEXP-hardness for both strict and lax semantics. For an overview of the results of this paper, see Table 1, and for an overview of the known complexity results from the literature, see Tables 4–6 on page 23.

The structure of the paper is as follows: in Section 2 basic definitions and properties of propositional inclusion logic are given. Section 3 concentrates on the model checking problem in propositional setting. In Section 4 we shift to basic definitions and properties of modal inclusion

TABLE 1 Overview of the complexity results proven in the article. Complexity classes refer to completeness results, '-h' denotes hardness.

	PInc		Minc	
Problem	Strict	Lax	Strict	Lax
Model checking	NP [Thm. 17]	P [Thm. 13]	NP [Thm. 29]	P [Thm. 28]
Validity	coNP [Thm. 9]	coNP [16]	coNEXP-h [Cor. 36]	coNEXP-h [Thm. 35]

logic. In Section 5 the model checking and validity problems of modal inclusion logic are studied. Section 6 concludes with some open problems and further directions of research.

This article is an extended version of the conference paper [17].

2 Propositional logics with team semantics

Let Φ be a finite nonempty set of proposition symbols. A function $s\colon \Phi \to \{0,1\}$ is called an *assignment*. A set X of assignments $s\colon \Phi \to \{0,1\}$ is called a *(propositional) team*. The set Φ is the *domain* of X. We denote by 2^{Φ} the set of *all assignments* $s\colon \Phi \to \{0,1\}$. If $\vec{p} = (p_1, \ldots, p_n)$ is a tuple of proposition symbols and s is an assignment, we write $s(\vec{p})$ for $(s(p_1), \ldots, s(p_n))$.

Let Φ be a set of proposition symbols. The syntax of classical propositional logic $\mathsf{PL}(\Phi)$ is given by the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi), \text{ where } p \in \Phi.$$

As usual, we often omit Φ and write PL instead of $\mathsf{PL}(\Phi)$.

We denote by $\models_{\mathsf{PL}}$ the ordinary satisfaction relation of classical propositional logic defined via assignments in the standard way. Next we give team semantics for the logic PL.

DEFINITION 1.1 (Lax team semantics).
Let Φ be a set of atomic propositions and let X be a team of domain Φ. The satisfaction relation $X \models \varphi$ for $\mathsf{PL}(\Phi)$-formulae φ is defined as follows:

$$\begin{aligned}
X \models p &\quad\Leftrightarrow\quad \forall s \in X : s(p) = 1,\\
X \models \neg p &\quad\Leftrightarrow\quad \forall s \in X : s(p) = 0.\\
X \models (\varphi \wedge \psi) &\quad\Leftrightarrow\quad X \models \varphi \text{ and } X \models \psi.\\
X \models (\varphi \vee \psi) &\quad\Leftrightarrow\quad Y \models \varphi \text{ and } Z \models \psi, \text{ for some } Y, Z \text{ such that } Y \cup Z = X.
\end{aligned}$$

The lax team semantics is considered to be the standard semantics for team-based logics. In this paper, we also consider a variant of team semantics called the *strict team semantics*. In strict team semantics, the above clause for disjunction is redefined as follows:

$$X \models_{\mathsf{str}} (\varphi \vee \psi) \;\Leftrightarrow\; Y \models \varphi \text{ and } Z \models \psi, \text{ for some } Y, Z \text{ such that } Y \cap Z = \emptyset \text{ and } Y \cup Z = X.$$

When L denotes a team-based propositional logic, we let $\mathsf{L}_{\mathsf{str}}$ denote the variant of the logic with strict semantics. Moreover, in order to improve readability, for strict semantics we use $\models_{\mathsf{str}}$ instead of $\models$. As a result lax semantics is used unless otherwise specified. The next proposition shows that the team semantics and the ordinary semantics for classical propositional logic defined via single assignments (denoted by $\models_{\mathsf{PL}}$) coincide. The proof is left to the reader.

PROPOSITION 2.2
Let φ be a PL-formula and let X be a propositional team. Then $X \models \varphi$ if and only if $\forall s \in X : s \models_{\mathsf{PL}} \varphi$.

The syntax of *propositional inclusion logic* $\mathsf{PInc}(\Phi)$ is obtained by extending the syntax of $\mathsf{PL}(\Phi)$ by the grammar rule

$$\varphi ::= \vec{p} \subseteq \vec{q},$$

	p	q	r	t
s_1	1	0	0	0
s_2	1	1	1	0
s_2'	1	1	1	1
s_3	0	1	0	0

FIGURE 1 Assignments for teams in Examples 2.5 and 2.8. Each row corresponds to a separate assignment s_i setting variables p, q, r, t to truth values 0/1.

where $\vec{p}$ and $\vec{q}$ are finite tuples of proposition symbols of the same length. The semantics for propositional inclusion atoms is defined as follows:

$$X \models \vec{p} \subseteq \vec{q} \quad \text{iff} \quad \forall s \in X \, \exists t \in X : s(\vec{p}) = t(\vec{q}).$$

REMARK
Extended propositional inclusion logic is the variant of PInc in which inclusion atoms of the form $\vec{\varphi} \subseteq \vec{\psi}$, where $\vec{\varphi}$ and $\vec{\psi}$ are tuples of PL-formulae, are allowed. Observe that this extension does not increase the complexity of the logic and on that account, in this paper, we only consider the non-extended variant.

Using a simple inductive argument, it is easy to see that the empty team satisfies every formula of PInc under both strict and lax semantics.

PROPOSITION 2.3 (Empty team property).
For every formula $\varphi \in \mathsf{PInc}$ it holds that $\emptyset \models \varphi$ and $\emptyset \models_{\mathsf{str}} \varphi$.

A logic L is *downward closed* if $X \models \varphi$ and $Y \subseteq X$ implies $Y \models \varphi$, for every formula $\varphi \in \mathsf{L}$ and teams X and Y. Note that PInc is not a downward closed logic. However, analogously to FO-inclusion-logic [9], satisfaction of PInc-formulae is closed under taking unions of satisfying teams.

PROPOSITION 2.4 (Closure under unions).
Let $\varphi \in \mathsf{PInc}$ and let X_i, for $i \in I$, be teams. Suppose that $X_i \models \varphi$ for each $i \in I$. Then $\bigcup_{i \in I} X_i \models \varphi$.

Similarly as in first-order team semantics [9], also for propositional logic the strict and the lax semantics coincide; meaning that $X \models \varphi$ iff $X \models_{\mathsf{str}} \varphi$ for all propositional teams X and all formulae $\varphi \in \mathsf{PL}$. However, this does not hold for propositional inclusion logic, for the following example shows that $\mathsf{PInc}_{\mathsf{str}}$ is not union closed. Moreover, we will show that the two different semantics lead to different complexities for the related model checking problems.

EXAMPLE 2.5
Let s_1, s_2 and s_3 be assignments as defined in Figure 1 and define $\varphi := \big(p \wedge (p \subseteq r)\big) \vee \big(q \wedge (q \subseteq r)\big)$. Note that $\{s_1, s_2\} \models_{\mathsf{str}} \varphi$ and $\{s_2, s_3\} \models_{\mathsf{str}} \varphi$, but $\{s_1, s_2, s_3\} \not\models_{\mathsf{str}} \varphi$.

However, $\mathsf{PInc}_{\mathsf{str}}$ satisfies a useful weaker form of union closure: it is straightforward to prove by induction on the formula structure that satisfaction of $\mathsf{PInc}_{\mathsf{str}}$-formulae is closed under unions of *singleton teams*.

LEMMA 2.6
Let X be a team and $\varphi \in \mathsf{PInc}_{\mathsf{str}}$. If $\{s\} \models_{\mathsf{str}} \varphi$ for every $s \in X$, then $X \models_{\mathsf{str}} \varphi$.

PROOF. The proof is by induction on φ. The cases other than the one for disjunctions are immediate. Assume then that $\{s\} \models_{\mathsf{str}} (\psi_1 \vee \psi_2)$ for every $s \in X$. By the semantics of the disjunction, for every $s \in X$ there is an $i \in \{0,1\}$ such that $\{s\} \models_{\mathsf{str}} \psi_i$. Let X_1 be the set of those assignments in X that satisfy φ_1, and define $X_2 := X \setminus X_1$. By induction hypothesis, $X_1 \models_{\mathsf{str}} \psi_1$ and $X_2 \models_{\mathsf{str}} \psi_2$. Consequently $X \models_{\mathsf{str}} (\psi_1 \vee \psi_2)$. □

One important difference between lax and strict semantics is that the former satisfies the following locality property, whereas the latter does not. A propositional logic L with team semantics satisfies *locality* if for every formula $\varphi \in \mathsf{L}$ and team X it holds that $X \models \varphi$ if and only if $X \upharpoonright \mathrm{Prop}(\varphi) \models \varphi$, where $\mathrm{Prop}(\varphi)$ is the set of proposition symbols that occur in φ, and $X \upharpoonright \mathrm{Prop}(\varphi)$ denotes the set of assignments obtained from X by restricting their domain to $\mathrm{Prop}(\varphi)$.

The fact that PInc satisfies locality is well known; the result follows by a straightforward inductive argument.

PROPOSITION 2.7
PInc satisfies locality.

The following example shows that in contrast $\mathsf{PInc}_{\mathsf{str}}$ does not satisfy locality.

EXAMPLE 2.8
Let s_1, s_2, s_2' and s_3 be assignments as defined in Figure 1 and φ as in Example 2.5. Note that $\{s_1, s_2, s_2', s_3\} \models_{\mathsf{str}} \varphi$ but $\{s_1, s_2, s_2', s_3\} \upharpoonright \{p, q, r\} \not\models_{\mathsf{str}} \varphi$.

3 Complexity of propositional inclusion logic

We now define the model checking, satisfiability and validity problems in the context of team semantics. Let L be a propositional logic with team semantics. A formula $\varphi \in \mathsf{L}$ is *satisfiable*, if there exists a nonempty[1] team X such that $X \models \varphi$. A formula $\varphi \in \mathsf{L}$ is *valid* if $X \models \varphi$ for all teams X such that the propositions in φ are in the domain of X. The satisfiability problem SAT(L) and the validity problem VAL(L) are defined in an obvious way: given a formula $\varphi \in \mathsf{L}$, decide whether the formula is satisfiable (valid, respectively). For the model checking problem MC(L) we consider combined complexity: given a formula $\varphi \in \mathsf{L}$ and a team X, decide whether $X \models \varphi$. See Table 2 for known complexity results for PL and PInc, together with partial results of this paper.

It was shown by Hannula *et al.* [16] that the validity problem of PInc is coNP-complete. Here we establish that the corresponding problem for $\mathsf{PInc}_{\mathsf{str}}$ is also coNP-complete. Our proof is similar to theirs, except that instead of union closure we use Lemma 2.6.

THEOREM 3.1
The validity problem for $\mathsf{PInc}_{\mathsf{str}}$ is coNP-complete w.r.t. $\leq_m^{\log}$.

PROOF. The coNP-hardness follows via Proposition 2.2 from the fact that the validity problem of PL is coNP-hard. On the other hand, by Lemma 2.6, a formula $\varphi \in \mathsf{PInc}_{\mathsf{str}}$ is valid if and only if it

[1] Many logics using team semantics have the empty team property and thus in the satisfiability problem it is customary to ask about the existence of a nonempty team.

TABLE 2 Complexity of the satisfiability, validity and model checking problems for propositional logics under both systems of semantics. The shown complexity classes refer to completeness results. Whenever strict and lax semantics coincide, only a single complexity result is shown, which is surrounded by dashes.

	Satisfiability		Validity		Model checking	
	Strict	Lax	Strict	Lax	Strict	Lax
PL	—— NP [5, 27] ——		—— coNP [5, 27] ——		—— NC^1 [2] ——	
PInc	EXP [19]	EXP [18]	coNP [Th. 3.1]	coNP [16]	NP [Th. 17]	P [Th. 3.5]

is satisfied by all singleton teams. It is easy to see that, over singleton teams, an inclusion atom ψ of the form

$$p_1, \ldots, p_n \subseteq q_1, \ldots, q_n$$

is equivalent to a PL-formula

$$\psi^* := \bigwedge_{1 \leq i \leq n} (p_i \wedge q_i) \vee (\neg p_i \wedge \neg q_i).$$

For each $\mathsf{PInc}_{\mathsf{str}}$-formula φ, let φ^* denote the PL-formula obtained from φ by substituting each inclusion atom ψ by its PL-translation ψ^*. Recall that on PL-formulae the strict and lax team semantics coincide, and thus it follows from Proposition 2.2 that φ^* is satisfied by every singleton team if and only if φ^* is valid with respect to $\models_{\mathsf{PL}}$. Since VAL(PL) is in coNP, the same is true for VAL($\mathsf{PInc}_{\mathsf{str}}$). □

3.1 Model checking in lax semantics is P-complete

In this section we construct a reduction from the monotone circuit value problem (MCVP) to the model checking problem of PInc. For an in-depth introduction to circuits, see Vollmer [42].

DEFINITION 3.2
A *monotone Boolean circuit* with n input gates and one output gate is a 3-tuple $C = (V, E, \alpha)$, where (V, E) is a finite, simple, directed, acyclic graph, and $\alpha \colon V \to \{\vee, \wedge, x_1, \ldots, x_n\}$ is a function such that the following conditions hold:

1. Every $v \in V$ has in-degree 0 or 2.
2. There exists exactly one $w \in V$ with out-degree 0. We call this node w the *output gate* of C and denote it by g_{out}.
3. If $v \in V$ is a node with in-degree 0, then $\alpha(v) \in \{x_1, \ldots, x_n\}$.
4. If $v \in V$ has in-degree 2, then $\alpha(v) \in \{\vee, \wedge\}$.
5. For each $1 \leq i \leq n$, there exists exactly one $v \in V$ with $\alpha(v) = x_i$.

Let $C = (V, E, \alpha)$ be a monotone Boolean circuit with n input gates and one output gate. Any sequence $b_1, \ldots, b_n \in \{0, 1\}$ of bits of length n is called an *input* to the circuit C. A function $\beta \colon V \to \{0, 1\}$ defined such that

$$\beta(v) := \begin{cases} b_i & \text{if } \alpha(v) = x_i, \\ \min\big(\beta(v_1), \beta(v_2)\big) & \text{if } \alpha(v) = \wedge, \text{ where } v_1 \neq v_2 \text{ and } (v_1, v), (v_2, v) \in E, \\ \max\big(\beta(v_1), \beta(v_2)\big) & \text{if } \alpha(v) = \vee, \text{ where } v_1 \neq v_2 \text{ and } (v_1, v), (v_2, v) \in E. \end{cases}$$

is called the *valuation of the circuit C under the input* $b_1, \ldots, b_n$. The *output of the circuit C* is then defined to be $\beta(g_{\text{out}})$.

The MCVP is the following decision problem: given a monotone circuit C and an input $b_1, \ldots, b_n \in \{0, 1\}$, is the output of the circuit 1?

PROPOSITION 3.3 ([12]).
MCVP is P-complete w.r.t. $\leq_m^{\log}$ reductions.

LEMMA 3.4
MC(PInc) under lax semantics is P-hard w.r.t. $\leq_m^{\log}$.

PROOF. We will establish a $\leq_m^{\log}$-reduction from MCVP to the model checking problem of PInc under lax semantics. Since MCVP is P-complete, the claim follows. More precisely, we will show how to construct, for each monotone Boolean circuit C with n input gates and for each input $\vec{b}$ for C, a team $X_{C,\vec{b}}$ and a PInc-formula φ_C such that

$$X_{C,\vec{b}} \models \varphi_C \text{ if and only if the output of the circuit } C \text{ with the input } \vec{b} \text{ is 1.}$$

We use teams to encode valuations of the circuit. For each gate v_i of a given circuit, we associate an assignment s_i. The crude idea is that if s_i is in the team under consideration, the value of the gate v_i with respect to the given input is 1. The formula φ_C is used to quantify a truth value for each Boolean gate of the circuit, and then for checking that the truth values of the gates propagate correctly. We next define the construction formally.

Let $C = (V, E, \alpha)$ be a monotone Boolean circuit with n input gates and one output gate and let $\vec{b} = (b_1 \ldots b_n) \in \{0, 1\}^n$ be an input to the circuit C. We stipulate that $V = \{v_0, \ldots, v_m\}$ and that v_0 is the output gate of C. Define

$$\tau_C := \{p_0, \ldots, p_m, p_\top, p_\bot\} \cup \{p_{k=i\vee j} \mid i < j, \alpha(v_k) = \vee, \text{ and } (v_i, v_k), (v_j, v_k) \in E\}.$$

For each $i \leq m$, we define the assignment $s_i \colon \tau_C \to \{0, 1\}$ as follows:

$$s_i(p) := \begin{cases} 1 & \text{if } p = p_i \text{ or } p = p_\top, \\ 1 & \text{if } p = p_{k=i\vee j} \text{ or } p = p_{k=j\vee i} \text{ for some } j, k \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, we define $s_\bot(p) = 1$ iff $p = p_\bot$ or $p = p_\top$. We note that the assignment $s_\bot$ will be the only assignment that maps $p_\bot$ to 1. We make use of the fact that for each gate v_i of C, we have that $s_\bot(p_i) = 0$. We define

$$X_{C,\vec{b}} := \{s_i \mid \alpha(v_i) \in \{\wedge, \vee\}\} \cup \{s_i \mid \alpha(v_i) \in \{x_i \mid b_i = 1\}\} \cup \{s_\bot\},$$

i.e. $X_{C,\vec{b}}$ consists of assignments for each of the Boolean gates, assignments for those input gates that are given 1 as an input, and of the auxiliary assignment $s_\bot$.

Let X be any nonempty subteam of $X_{C,\vec{b}}$ such that $s_\bot \in X$. We have

$$\begin{aligned} X &\models p_\top \subseteq p_0 && \text{iff } s_0 \in X \\ X &\models p_i \subseteq p_j && \text{iff } (s_i \in X \text{ implies } s_j \in X) \\ X &\models p_k \subseteq p_{k=i\vee j} && \text{iff } (i<j, (v_i,v_k),(v_j,v_k) \in E, \alpha(v_k)=\vee \\ & && \quad \text{and } s_k \in X \text{ implies that } s_i \in X \text{ or } s_j \in X). \end{aligned} \tag{2}$$

Recall the intuition that $s_i \in X$ should hold iff the value of the gate v_i is 1. Define

$$\begin{aligned} \psi_{\text{out=1}} &:= p_\top \subseteq p_0, \\ \psi_\wedge &:= \bigwedge\{p_i \subseteq p_j \mid (v_j,v_i) \in E \text{ and } \alpha(p_i)=\wedge\}, \\ \psi_\vee &:= \bigwedge\{p_k \subseteq p_{k=i\vee j} \mid i<j, (v_i,v_k)\in E, (v_j,v_k)\in E, \text{ and } \alpha(v_k)=\vee\}, \\ \varphi_C &:= \neg p_\bot \vee (\psi_{\text{out=1}} \wedge \psi_\wedge \wedge \psi_\vee). \end{aligned}$$

We claim that $X_{C,\vec{b}} \models \varphi_C$ iff the output of C with the input $\vec{b}$ is 1.

The idea of the reduction is the following: the disjunction in ϕ_C is used to guess a team Y for the right disjunct that encodes the valuation β of the circuit C. The right disjunct is then evaluated with respect to the team Y with the intended meaning that $\beta(v_i)=1$ whenever $s_i \in Y$. Note that Y is always as required in (2). The formula $\psi_{\text{out=1}}$ is used to state that $\beta(v_0)=1$, whereas the formulae $\psi_\wedge$ and $\psi_\vee$ are used to propagate the truth value 1 down the circuit. The assignment $s_\bot$ and the proposition $p_\bot$ are used as an auxiliary to make sure that Y is nonempty and to deal with the propagation of the value 0 by the subformulae of the form $p_i \subseteq p_j$.

Now observe that the team $X_{C,\vec{b}}$ can be easily computed by a logspace Turing machine that scans the input for $\wedge$-gates, $\vee$-gates and true input gates, and then outputs the corresponding team members s_i in a bitwise fashion. The formula φ_C can be computed in logspace as well:

1. the left disjunct does not depend on the input,
2. for $\psi_\wedge$ we only need to scan for the $\wedge$-gates and output the inclusion-formulae for the corresponding edges,
3. for $\psi_\vee$ we need to maintain two binary counters for i and j, and use them for searching for those disjunction gates that satisfy $i<j$.

Consequently, the reduction can be computed in logspace. □

For the proof of the above lemma it is not important that lax semantics is considered; the same proof works also for the strict semantics. However, as we will show in the next section, we can establish a stronger result for the model checking problem of $\mathsf{PInc}_{\mathsf{str}}$; namely that it is NP-hard. In Section 5.1 we will show that the model checking problem for modal inclusion logic with lax semantics is in P (Lemma 5.2). Since PInc is essentially a fragment of this logic, by combining Lemmas 3.4 and 5.2, we obtain the following theorem.

THEOREM 3.5
MC(PInc) under lax semantics is P-complete w.r.t. $\leq_m^{\log}$.

3.2 Model checking in strict semantics is NP-complete

In this section we reduce the set splitting problem, a well-known NP-complete problem, to the model checking problem of $\mathsf{PInc}_{\mathsf{str}}$.

DEFINITION 3.6
The *set splitting* problem is the following decision problem:

Input: A family $\mathcal{F}$ of subsets of a finite set S.
Problem: Do there exist subsets S_1 and S_2 of S such that
1. S_1 and S_2 are a partition of S (i.e. $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$),
2. for each $A \in \mathcal{F}$, there exist $a_1, a_2 \in A$ such that $a_1 \in S_1$ and $a_2 \in S_2$?

PROPOSITION 3.7 ([11]).
The set splitting problem is NP-complete w.r.t. $\leq_m^{\log}$.

The following proof relies on the fact that strict semantics is considered. It cannot hold for lax semantics unless $\mathsf{P} = \mathsf{NP}$.

LEMMA 3.8
$\mathrm{MC}(\mathsf{PInc}_{\mathsf{str}})$ is NP-hard with respect to $\leq_m^{\log}$.

PROOF. We give a reduction from the set splitting problem to the model checking problem of PInc under strict semantics.

Let $\mathcal{F}$ be an instance of the set splitting problem. We stipulate that $\mathcal{F} = \{B_1, \ldots, B_n\}$ and that $\bigcup \mathcal{F} = \{a_1, \ldots, a_k\}$, where $n, k \in \mathbb{N}$. We will introduce fresh propositions p_i and q_j for each point $a_i \in \bigcup \mathcal{F}$ and set $B_j \in \mathcal{F}$. We will then encode the family of sets $\mathcal{F}$ by assignments over these propositions; each assignment s_i will correspond to a unique point a_i. Formally, let $\tau_{\mathcal{F}}$ denote the set $\{p_1, \ldots, p_k, q_1, \ldots, q_n, p_\top, p_c, p_d\}$ of propositions. For each $i \in \{1, \ldots, k\}$, we define the assignment $s_i \colon \tau_{\mathcal{F}} \to \{0, 1\}$ as follows:

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

We also define two auxiliary assignments $s_c \colon \tau_{\mathcal{F}} \to \{0, 1\}$ and $s_d \colon \tau_{\mathcal{F}} \to \{0, 1\}$ as follows: s_c maps $p_\top$ and p_c to 1 and everything else to 0, whereas s_d maps $p_\top$ and p_d to 1 and everything else to 0.

Define $X_{\mathcal{F}} := \{s_1, \ldots, s_k, s_c, s_d\}$, i.e. $X_{\mathcal{F}}$ consists of assignments s_i corresponding to each of the points $a_i \in \bigcup \mathcal{F}$ and of two auxiliary assignments s_c and s_d. Note that the only assignment in $X_{\mathcal{F}}$ that maps p_c (p_d, resp.) to 1 is s_c (s_d, resp.) and that every assignment maps $p_\top$ to 1. Moreover, note that for $1 \leq i \leq k$ and $1 \leq j \leq n$, $s_i(q_j) = 1$ iff $a_i \in B_j$. Now define

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

We claim that $X_{\mathcal{F}} \models_{\mathsf{str}} \varphi_{\mathcal{F}}$ iff the output of the set splitting problem with input $\mathcal{F}$ is 'yes'.

The proof is straightforward. Note that $X_{\mathcal{F}} \models_{\mathsf{str}} \varphi_{\mathcal{F}}$ iff $X_{\mathcal{F}}$ can be partitioned into two subteams Y_1 and Y_2 such that

$$Y_1 \models_{\mathsf{str}} \neg p_c \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i \text{ and } Y_2 \models_{\mathsf{str}} \neg p_d \wedge \bigwedge_{i \leq n} p_\top \subseteq q_i.$$

Teams Y_1 and Y_2 are both nonempty, since $s_d \in Y_1$ and $s_c \in Y_2$. Also, for a nonempty subteam Y of $X_{\mathcal{F}}$, we have that $Y \models_{\mathsf{str}} p_\top \subseteq q_j$ iff there exists $s_i \in Y$ such that $s_i(q_j) = 1$, or equivalently, $a_i \in B_j$.

It is now evident that if $X_{\mathcal{F}} \models_{\mathsf{str}} \varphi_{\mathcal{F}}$ then the related subteams Y_1 and Y_2 directly construct a positive answer to the set splitting problem. Likewise, any positive answer to the set splitting problem can be used to directly construct the related subteams Y_1 and Y_2.

In order to compute the assignments s_i and by this the team $X_{\mathcal{F}}$ on a logspace machine, we need to implement two binary counters to count through $1 \leq i \leq k$ for the propositions p_i and $1 \leq j \leq n$ for the propositions q_i. The formula $\varphi_{\mathcal{F}}$ is constructed in logspace by simply outputting it step by step with the help of a binary counter for the interval $1 \leq i \leq n$. As a result the whole reduction can be implemented on a logspace Turing machine. □

In Section 5.1, we establish that the model checking problem of modal inclusion logic with strict semantics is in NP (Theorem 5.5). Since $\mathsf{PInc}_{\mathsf{str}}$ is essentially a fragment of this logic, together with Lemma 3.8, we obtain the following theorem.

THEOREM 3.9
MC($\mathsf{PInc}_{\mathsf{str}}$) is NP-complete with respect to $\leq_m^{\log}$.

4 Modal logics with team semantics

Let Φ be a set of proposition symbols. The syntax of basic modal logic $\mathsf{ML}(\Phi)$ is generated by the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \Diamond\varphi \mid \Box\varphi, \text{where } p \in \Phi.$$

By $\varphi^\perp$ we denote the formula that is obtained from $\neg\varphi$ by pushing all negation symbols to the atomic level using the standard duality between $\wedge$ ($\Box$) and $\vee$ ($\Diamond$). A (Kripke) *Φ-model* is a tuple $\mathfrak{M} = (W, R, V)$, where W, called the *domain* of $\mathfrak{M}$, is a nonempty set, $R \subseteq W \times W$ is a binary relation, and $V\colon \Phi \to \mathcal{P}(W)$ is a valuation of the proposition symbols. By $\models_{\mathsf{ML}}$ we denote the *satisfaction relation* of basic modal logic that is defined via pointed *Φ-models* in the standard way. Any subset T of the domain of a Kripke model $\mathfrak{M}$ is called a *(modal) team of* $\mathfrak{M}$. Before we define *team semantics* for ML, we introduce some auxiliary notations.

DEFINITION 4.1
Let $\mathfrak{M} = (W, R, V)$ be a model and T and S teams of $\mathfrak{M}$. Define that

$$R[T] := \{w \in W \mid \exists v \in T \text{ s.t. } vRw\} \text{ and } R^{-1}[T] := \{w \in W \mid \exists v \in T \text{ s.t. } wRv\}.$$

For teams T and S of $\mathfrak{M}$, we write $T[R]S$ if $S \subseteq R[T]$ and $T \subseteq R^{-1}[S]$.

Accordingly, $T[R]S$ if and only if for every $w \in T$, there exists some $v \in S$ such that wRv, and for every $v \in S$, there exists some $w \in T$ such that wRv. We are now ready to define team semantics for ML.

DEFINITION 4.2 (Lax team semantics).
Let $\mathfrak{M}$ be a Kripke model and T a team of $\mathfrak{M}$. The satisfaction relation $\mathfrak{M}, T \models \varphi$ for $\mathsf{ML}(\Phi)$ is

defined as follows.

$$\begin{aligned}
\mathfrak{M}, T \models p &\quad\Leftrightarrow\quad w \in V(p) \text{ for every } w \in T.\\
\mathfrak{M}, T \models \neg p &\quad\Leftrightarrow\quad w \notin V(p) \text{ for every } w \in T.\\
\mathfrak{M}, T \models (\varphi \wedge \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T \models \varphi \text{ and } \mathfrak{M}, T \models \psi.\\
\mathfrak{M}, T \models (\varphi \vee \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T_1 \models \varphi \text{ and } \mathfrak{M}, T_2 \models \psi \text{ for some } T_1 \text{ and } T_2 \text{ s.t. } T_1 \cup T_2 = T.\\
\mathfrak{M}, T \models \Diamond\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, T' \models \varphi \text{ for some } T' \text{ s.t. } T[R]T'.\\
\mathfrak{M}, T \models \Box\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, T' \models \varphi, \text{ where } T' = R[T].
\end{aligned}$$

Analogously to the propositional case, we also consider the *strict* variant of team semantics for modal logic. In the *strict* team semantics, we have the following alternative semantic definitions for the disjunction and diamond (where W denotes the domain of $\mathfrak{M}$).

$$\begin{aligned}
\mathfrak{M}, T \models_{\mathsf{str}} (\varphi \vee \psi) &\quad\Leftrightarrow\quad \mathfrak{M}, T_1 \models \varphi \text{ and } \mathfrak{M}, T_2 \models \psi\\
&\qquad\qquad \text{for some } T_1 \text{ and } T_2 \text{ such that } T_1 \cup T_2 = T \text{ and } T_1 \cap T_2 = \emptyset.\\
\mathfrak{M}, T \models_{\mathsf{str}} \Diamond\varphi &\quad\Leftrightarrow\quad \mathfrak{M}, f[T] \models \varphi \text{ for some } f : T \to W \text{ s.t. } \forall w \in T : wRf(w).\\
&\qquad\qquad \text{Here } f[T] \text{ denotes the image of } T \text{ under } f.
\end{aligned}$$

When L is a team-based modal logic, we let $\mathsf{L}_{\mathsf{str}}$ denote its variant with strict semantics. As in the propositional case, for strict semantics we use $\models_{\mathsf{str}}$ instead of $\models$. The formulae of ML have the following flatness property, independent of the considered semantics (see, e.g. the survey of Durand *et al.* [7]). The proof is left to the reader.

PROPOSITION 4.3
Let $\mathfrak{M}$ be a Kripke model and T be a team of $\mathfrak{M}$. Then, for every formula φ of $\mathsf{ML}(\Phi)$: $\mathfrak{M}, T \models \varphi \Leftrightarrow \forall w \in T : \mathfrak{M}, w \models_{\mathsf{ML}} \varphi \Leftrightarrow \mathfrak{M}, T \models_{\mathsf{str}} \varphi$.

Note, that the law of excluded middle for modal formulae follows from the flatness property.

The syntax of *modal inclusion logic* $\mathsf{Minc}(\Phi)$ and *extended modal inclusion logic* $\mathsf{EMinc}(\Phi)$ is obtained by extending the syntax of $\mathsf{ML}(\Phi)$ by the following grammar rule for each $n \in \mathbb{N}$:

$$\varphi ::= (\varphi_1, \ldots, \varphi_n) \subseteq (\psi_1, \ldots, \psi_n),$$

where $\varphi_1, \psi_1, \ldots, \varphi_n, \psi_n \in \mathsf{ML}(\Phi)$. Additionally, for $\mathsf{Minc}(\Phi)$, we require that $\varphi_1, \psi_1, \ldots, \varphi_n, \psi_n$ are proposition symbols. The semantics for these inclusion atoms is defined as follows:

$$\begin{aligned}
&\mathfrak{M}, T \models (\varphi_1, \ldots, \varphi_n) \subseteq (\psi_1, \ldots, \psi_n) \Leftrightarrow\\
&\qquad \forall w \in T \exists v \in T : (\mathfrak{M}, \{w\} \models \varphi_i \Leftrightarrow \mathfrak{M}, \{v\} \models \psi_i) \text{ is true for every } 1 \leq i \leq n.
\end{aligned}$$

Using a simple inductive argument, it is easy to see that the empty team satisfies every formula of Minc under both strict and lax semantics.

PROPOSITION 4.4 (Empty team property).
For every formula $\varphi \in \mathsf{Minc}$ and Kripke model $\mathfrak{M}$ it holds that $\mathfrak{M}, \emptyset \models \varphi$ and $\mathfrak{M}, \emptyset \models_{\mathsf{str}} \varphi$.

The following proposition is proven in the same way as the analogous results for first-order inclusion logic [9]. A modal logic L is union closed if $\mathfrak{M}, T \models \varphi$ and $\mathfrak{M}, S \models \varphi$ implies that $\mathfrak{M}, T \cup S \models \varphi$, for every $\varphi \in \mathsf{L}$.

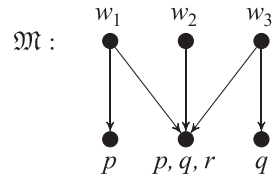

FIGURE 2 Kripke model for Example 4.7. The upper worlds have the name w_i, whereas for the lower worlds the symbols p, q, r denote which propositions are labelled in that respective world.

PROPOSITION 4.5 (Union Closure).
The logics ML, Minc, EMinc are union closed.

Analogously to the propositional case, it is easy to see that, for ML-formulae, the strict and lax semantics coincide. Again, as in the propositional case, this does not hold for Minc or EMinc. Note that since $\mathsf{PInc}_{\mathsf{str}}$ is not union closed, neither is $\mathsf{Minc}_{\mathsf{str}}$ nor $\mathsf{EMinc}_{\mathsf{str}}$.

A modal logic L with team semantics satisfies *locality* if for every set of proposition symbols Φ, formula $\varphi \in \mathsf{L}(\Phi)$, Kripke structure $\mathfrak{M} = (W, R, V)$ and team T of $\mathfrak{M}$ it holds that $\mathfrak{M}, T \models \varphi$ if and only if $\mathfrak{M}', T \models \varphi$, where $\mathfrak{M}' = (W, R, V')$ and $V' = V \upharpoonright \mathrm{Prop}(\varphi)$.

In contrast to the propositional case, $\mathsf{Minc}_{\mathsf{str}}$ and $\mathsf{EMinc}_{\mathsf{str}}$ satisfies locality. This holds, since insisting locality in modal setting does not yield any collapse of the size of the team under consideration; this kind of collapse is the real culprit of Example 2.8. The following proposition follows by a straightforward inductive argument

PROPOSITION 4.6
Minc, EMinc, $\mathsf{Minc}_{\mathsf{str}}$ and $\mathsf{EMinc}_{\mathsf{str}}$ all satisfy locality.

In contrary to the propositional case, Lemma 2.6 fails in the modal case as the following example illustrates.

EXAMPLE 4.7
Let $\mathfrak{M}$ be as depicted in Figure 2 and define

$$\varphi := \Box\Big(\big(p \wedge (p \subseteq r)\big) \vee \big(q \wedge (q \subseteq r)\big)\Big).$$

Now $\mathfrak{M}, \{w_i\} \models_{\mathsf{str}} \varphi$, for $i \in \{1, 2, 3\}$, but $\mathfrak{M}, \{w_1, w_2, w_3\} \not\models_{\mathsf{str}} \varphi$.

5 Model checking and validity in modal team semantics

The model checking, satisfiability and validity problems in the context of team semantics of modal logic are defined analogously to the propositional case. Let $\mathsf{L}(\Phi)$ be a modal logic with team semantics. A formula $\varphi \in \mathsf{L}(\Phi)$ is *satisfiable*, if there exists a Kripke Φ-model $\mathfrak{M}$ and a nonempty[2] team T of $\mathfrak{M}$ such that $\mathfrak{M}, T \models \varphi$. A formula $\varphi \in \mathsf{L}(\Phi)$ is *valid*, if $\mathfrak{M}, T \models \varphi$ for every Φ-model $\mathfrak{M}$ and every team T of $\mathfrak{M}$. The satisfiability problem SAT(L) and the validity problem VAL(L) are defined in the obvious way: given a formula $\varphi \in \mathsf{L}$, decide whether the formula is satisfiable (valid, respectively). For model checking MC(L) we consider combined complexity: given a formula

[2]Many logics using team semantics have the empty team property and thus in the satisfiability problem it is customary to ask about the existence of a nonempty team.

TABLE 3 Complexity of satisfiability, validity and model checking for modal logics under both strict and lax semantics. The given complexity classes refer to completeness results and '-h.' denotes hardness. The complexities for Minc and EMinc coincide, see Theorems 5.4, 5.5 and 5.13. Whenever strict and lax semantics coincide, only a single complexity result is shown, which is surrounded by dashes.

	Satisfiability		Validity		Model checking	
	Strict	Lax	Strict	Lax	Strict	Lax
ML	— PSPACE [26]—		——— PSPACE [26] ———		——— P [4, 36] ———	
Minc	EXP [19]	EXP [18]	coNEXP-h. [C. 5.12]	coNEXP-h. [Th. 5.11]	NP [Th. 5.5]	P [Th. 5.4]

$\varphi \in \mathsf{L}$, a Kripke model $\mathfrak{M}$ and a team T of $\mathfrak{M}$, decide whether $\mathfrak{M}, T \models \varphi$. See Table 3 for known complexity results on ML and Minc, together with partial results of this paper.

5.1 Complexity of model checking

Let $\mathfrak{M}$ be a Kripke model, T be a team of $\mathfrak{M}$ and φ be a formula of Minc. By maxsub(T, φ), we denote the maximum subteam T' of T such that $\mathfrak{M}, T' \models \varphi$. Since Minc is union closed (by Proposition 4.5), such a maximum subteam always exists.

LEMMA 5.1
If φ is a proposition symbol, its negation, or an inclusion atom, then maxsub(T, φ) can be computed in polynomial time with respect to $|T| + |\varphi|$.

PROOF. If φ is a proposition symbol or its negation, the claim follows from flatness in a straightforward way. Assume then that $T = \{w_1, \ldots w_n\}$ and $\varphi = p_1, \ldots, p_k \subseteq q_1, \ldots, q_k$. Let $G = (V, E)$ be a directed graph such that $V = T$ and $(u, v) \in E$ iff the value of p_i in u is the same as the value of q_i in v, for each $1 \leq i \leq k$.

The graph G describes the inclusion dependencies between the points in the following sense: if $w \in$ maxsub(T, φ), then there exists some $v \in$ maxsub(T, φ) such that $(w, v) \in E$. Clearly G can be computed in time $\mathcal{O}(n^2 k)$. In order to construct maxsub(T, φ), we, round by round, delete all vertices from G with out-degree 0. Formally, we define a sequence $G_0, \ldots, G_n$ of graphs recursively. We define that $G_0 := G$ and that G_{j+1} is the graph obtained from G_j by deleting all of those vertices from G_j that have out-degree 0 in G_j. Let i be the smallest integer such that $G_i = (V_i, E_i)$ has no vertices of out-degree 0. Clearly $i \leq n$, and moreover, G_i is computable from G in time $\mathcal{O}(n^3)$. It is easy to check that $V_i =$ maxsub(T, φ). □

For the following lemma it is crucial that lax semantics is considered. The lemma cannot hold for strict semantics unless P = NP.

LEMMA 5.2
MC(Minc) under lax semantics is in P.

PROOF. We will present a labelling algorithm for model checking $\mathfrak{M}, T \models \varphi$. Let subOcc$(\varphi)$ denote the set of all *occurrences* of subformulae of φ. Below we denote occurrences as if they were formulae, but we actually refer to some particular occurrence of the formula.

A function f : subOcc$(\varphi) \to \mathcal{P}(W)$ is called a labelling function of φ in $\mathfrak{M}$. We will next give an algorithm for computing a sequence $f_0, f_1, f_2, \ldots$, of such labelling functions.

- Define $f_0(\psi) = W$ for each $\psi \in \mathrm{subOcc}(\varphi)$.
- For odd $i \in \mathbb{N}$, define $f_i(\psi)$ bottom up as follows:
  1. For literal ψ, define $f_i(\psi) := \mathrm{maxsub}(f_{i-1}(\psi), \psi)$.
  2. $f_i(\psi \wedge \theta) := f_i(\psi) \cap f_i(\theta)$.
  3. $f_i(\psi \vee \theta) := f_i(\psi) \cup f_i(\theta)$.
  4. $f_i(\Diamond\psi) := \{w \in f_{i-1}(\Diamond\psi) \mid R[w] \cap f_i(\psi) \neq \emptyset\}$.
  5. $f_i(\Box\psi) := \{w \in f_{i-1}(\Box\psi) \mid R[w] \subseteq f_i(\psi)\}$.
- For even $i \in \mathbb{N}$ larger than 0, define $f_i(\psi)$ top to bottom as follows:
  1. Define $f_i(\varphi) := f_{i-1}(\varphi) \cap T$.
  2. If $\psi = \theta \wedge \gamma$, define $f_i(\theta) := f_i(\gamma) := f_i(\theta \wedge \gamma)$.
  3. If $\psi = \theta \vee \gamma$, define $f_i(\theta) := f_{i-1}(\theta) \cap f_i(\theta \vee \gamma)$ and $f_i(\gamma) := f_{i-1}(\gamma) \cap f_i(\theta \vee \gamma)$.
  4. If $\psi = \Diamond\theta$, define $f_i(\theta) := f_{i-1}(\theta) \cap R[f_i(\Diamond\theta)]$.
  5. If $\psi = \Box\theta$, define $f_i(\theta) := f_{i-1}(\theta) \cap R[f_i(\Box\theta)]$.

By a straightforward induction on i, we can prove that $f_{i+1}(\psi) \subseteq f_i(\psi)$ for every $\psi \in \mathrm{subOcc}(\varphi)$. The only nontrivial induction step is that for $f_{i+1}(\theta)$ and $f_{i+1}(\gamma)$, when $i+1$ is even and $\psi = \theta \wedge \gamma$. To deal with this step, observe that, by the definition of f_{i+1} and f_i, we have $f_{i+1}(\theta) = f_{i+1}(\gamma) = f_{i+1}(\psi)$ and $f_i(\psi) \subseteq f_i(\theta), f_i(\gamma)$. Note also that for even $i+1$ the direction of the proof is from larger formulae to subformulae; in particular we may assume that $f_{i+1}(\psi) \subseteq f_i(\psi)$. Now by connecting the previous observations, we obtain that $f_{i+1}(\theta) = f_{i+1}(\psi) \subseteq f_i(\psi) \subseteq f_i(\theta)$ and $f_{i+1}(\gamma) = f_{i+1}(\psi) \subseteq f_i(\psi) \subseteq f_i(\gamma)$.

It follows that there is an integer $j \leq 2 \cdot |W| \cdot |\varphi|$ such that $f_{j+2} = f_{j+1} = f_j$. We denote this fixed point f_j of the sequence $f_0, f_1, f_2, \ldots$ by f_∞. By Lemma 5.1 the outcome of $\mathrm{maxsub}(\cdot,\cdot)$ is computable in polynomial time with respect to its input. That being, clearly f_{i+1} can be computed from f_i in polynomial time with respect to $|W| + |\varphi|$. On that account f_∞ is also computable in polynomial time with respect to $|W| + |\varphi|$.

We will next prove by induction on $\psi \in \mathrm{subOcc}(\varphi)$ that $\mathfrak{M}, f_\infty(\psi) \models \psi$. Note first that there is an odd integer i and an even integer j such that $f_\infty = f_i = f_j$.

1. If ψ is a literal, the claim is true since $f_\infty = f_i$ and $f_i(\psi) = \mathrm{maxsub}(f_{i-1}(\psi), \psi)$.
2. Assume next that $\psi = \theta \wedge \gamma$, and the claim is true for θ and γ. Since $f_\infty = f_j$, we have $f_\infty(\psi) = f_\infty(\theta) = f_\infty(\gamma)$, as a result, by induction hypothesis, $\mathfrak{M}, f_\infty(\psi) \models \theta \wedge \gamma$.
3. In the case $\psi = \theta \vee \gamma$, we obtain the claim $\mathfrak{M}, f_\infty(\psi) \models \psi$ by using the induction hypothesis, and the observation that $f_\infty(\psi) = f_i(\psi) = f_i(\theta) \cup f_i(\gamma) = f_\infty(\theta) \cup f_\infty(\gamma)$.
4. Assume then that $\psi = \Diamond\theta$. Since $f_\infty = f_i$, we have $f_\infty(\psi) = \{w \in f_{i-1}(\psi) \mid R[w] \cap f_\infty(\theta) \neq \emptyset\}$, as a consequence $f_\infty(\psi) \subseteq R^{-1}[f_\infty(\theta)]$. On the other hand, since $f_\infty = f_j$, we have $f_\infty(\theta) = f_{j-1}(\theta) \cap R[f_\infty(\psi)]$, for this reason $f_\infty(\theta) \subseteq R[f_\infty(\psi)]$. Thus $f_\infty(\psi)[R] f_\infty(\theta)$, and using the induction hypothesis, we see that $\mathfrak{M}, f_\infty(\psi) \models \psi$.
5. Assume finally that $\psi = \Box\theta$. Since $f_\infty = f_i$, we have $R[f_\infty(\psi)] \subseteq f_\infty(\theta)$. On the other hand, since $f_\infty = f_j$, we have $f_\infty(\theta) \subseteq R[f_\infty(\psi)]$. This shows that $f_\infty(\theta) = R[f_\infty(\psi)]$, that being the case by the induction hypothesis, $\mathfrak{M}, f_\infty(\psi) \models \psi$.

In particular, if $f_\infty(\varphi) = T$, then $\mathfrak{M}, T \models \varphi$. Consequently, to complete the proof of the lemma, it suffices to prove that the converse implication is true, as well. To prove this, assume that $\mathfrak{M}, T \models \varphi$. Then for each $\psi \in \mathrm{subOcc}(\varphi)$, there is a team T_ψ such that

1. $T_\varphi = T$.
2. If $\psi = \theta \wedge \gamma$, then $T_\psi = T_\theta = T_\gamma$.
3. If $\psi = \theta \vee \gamma$, then $T_\psi = T_\theta \cup T_\gamma$.

4. If $\psi = \Diamond\theta$, then $T_\psi[R]T_\theta$.
5. If $\psi = \Box\theta$, then $T_\theta = R[T_\psi]$.
6. If ψ is a literal, then $\mathfrak{M}, T_\psi \models \psi$.

We prove by induction on i that $T_\psi \subseteq f_i(\psi)$ for all $\psi \in \mathrm{subOcc}(\varphi)$. For $i = 0$, this is obvious, since $f_0(\psi) = W$ for all ψ. Assume next that $i + 1$ is odd and the claim is true for i. We prove the claim $T_\psi \subseteq f_i(\psi)$ by induction on ψ.

1. If ψ is a literal, then $f_{i+1}(\psi) = \mathrm{maxsub}(f_i(\psi), \psi)$. Since $\mathfrak{M}, T_\psi \models \psi$, and by induction hypothesis, $T_\psi \subseteq f_i(\psi)$, the claim $T_\psi \subseteq f_{i+1}(\psi)$ is true.
2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on θ and γ, we have $T_\psi = T_\theta \subseteq f_{i+1}(\theta)$ and $T_\psi = T_\gamma \subseteq f_{i+1}(\gamma)$. For this reason, we get $T_\psi \subseteq f_{i+1}(\theta) \cap f_{i+1}(\gamma) = f_{i+1}(\psi)$.
3. The case $\psi = \theta \vee \gamma$ is similar to the previous one; we omit the details.
4. If $\psi = \Diamond\theta$, then $f_{i+1}(\psi) = \{w \in f_i(\psi) \mid R[w] \cap f_{i+1}(\theta) \neq \emptyset\}$. By the two induction hypotheses on i and θ, we have $\{w \in T_\psi \mid R[w] \cap T_\theta \neq \emptyset\} \subseteq f_{i+1}(\psi)$. The claim follows from this, since the condition $R[w] \cap T_\theta \neq \emptyset$ is true for all $w \in T_\psi$.
5. The case $\psi = \Box\theta$ is again similar to the previous one, so we omit the details.

Assume then that $i + 1$ is even and the claim is true for i. This time we prove the claim $T_\psi \subseteq f_i(\psi)$ by top to bottom induction on ψ.

1. By assumption, $T_\varphi = T$, whence by induction hypothesis, $T_\varphi \subseteq f_i(\varphi) \cap T = f_{i+1}(\varphi)$.
2. Assume that $\psi = \theta \wedge \gamma$. By induction hypothesis on ψ, we have $T_\psi \subseteq f_{i+1}(\psi)$. Since $T_\psi = T_\theta = T_\gamma$ and $f_{i+1}(\psi) = f_{i+1}(\theta) = f_{i+1}(\gamma)$, this implies that $T_\theta \subseteq f_{i+1}(\theta)$ and $T_\gamma \subseteq f_{i+1}(\gamma)$.
3. Assume that $\psi = \theta \vee \gamma$. Using the fact that $T_\theta \subseteq T_\psi$, and the two induction hypotheses on i and ψ, we see that $T_\theta \subseteq f_i(\theta) \cap T_\psi \subseteq f_i(\theta) \cap f_{i+1}(\psi) = f_{i+1}(\theta)$. Similarly, we see that $T_\gamma \subseteq f_{i+1}(\gamma)$.
4. Assume that $\psi = \Diamond\theta$. By the induction hypothesis on i, we have $T_\theta \subseteq f_i(\theta)$, and by the induction hypothesis on ψ, we have $T_\theta \subseteq R[T_\psi] \subseteq R[f_{i+1}(\psi)]$. Accordingly, we see that $T_\theta \subseteq f_i(\theta) \cap R[f_{i+1}(\psi)] = f_{i+1}(\theta)$.
5. The case $\psi = \Box\theta$ is similar to the previous one; we omit the details.

It follows now that $T = T_\varphi \subseteq f_\infty(\varphi)$. Since $f_\infty(\varphi) \subseteq f_2(\varphi) \subseteq T$, we conclude that $f_\infty(\varphi) = T$. This completes the proof of the implication $\mathfrak{M}, T \models \varphi \Rightarrow f_\infty(\varphi) = T$. □

The following lemma then follows, since in the context of model checking, we may replace modal formulae that appear as parameters in inclusion atoms by fresh proposition symbols with the same extension.

LEMMA 5.3
MC(EMinc) under lax semantics is in P.

PROOF. The result follows by a polynomial time reduction to the model checking problem of Minc: let $(W, R, V), T$ be a team pointed Kripke model and φ be a formula of EMinc. Let $\varphi_1, \ldots, \varphi_n$ be exactly those subformulae of φ that occur as a parameter of some inclusion atom in φ and let $p_1, \ldots, p_n$ be distinct fresh proposition symbols. Let V' be a valuation defined as follows:

$$V'(p) := \begin{cases} \{w \in W \mid (W, R, V), w \models_{\mathsf{ML}} \varphi_i\} & \text{if } p = p_i, \\ V(p) & \text{otherwise.} \end{cases}$$

Let φ^* denote the formula obtained from φ by simultaneously substituting each φ_i by p_i. It is easy to check that $(W, R, V), T \models \varphi$ if and only if $(W, R, V'), T \models \varphi^*$. Moreover, φ^* can be clearly

computed from φ in polynomial time. Likewise, V' can be computed in polynomial time; since each φ_i is a modal formula, the truth set of that formula in (W, R, V) can be computed in polynomial time by the standard labelling algorithm used in modal logic (see e.g. [1]), and the number of such computations is bounded above by the size of φ. As a consequence the result follows from Lemma 5.1. □

By combining Lemmas 3.4, 5.2 and 5.3, we obtain the following theorem.

THEOREM 5.4
MC(Minc) and MC(EMinc) are P-complete w.r.t. $\leq_m^{\log}$.

THEOREM 5.5
MC($\mathsf{Minc}_{\mathsf{str}}$) and MC($\mathsf{EMinc}_{\mathsf{str}}$) are NP-complete w.r.t. $\leq_m^{\log}$.

PROOF. The NP-hardness follows from the propositional case, i.e. from Lemma 3.8.

In order to establish inclusion, we note that the obvious brute force algorithm for model checking for EMinc works in NP: for disjunctions and diamonds, we use nondeterminism to guess the correct partitions or successor teams, respectively. Conjunctions are dealt with sequentially and for boxes the unique successor team can be computed by brute force in quadratic time. Checking whether a team satisfies an inclusion atom or a (negated) proposition symbol can be computed by brute force in polynomial time (this also follows directly from Lemma 5.1). □

5.2 Dependency quantifier Boolean formulae

Deciding whether a given quantified Boolean formula (qBf) is valid is a canonical PSPACE-complete problem. *Dependency quantifier Boolean formulae* introduced by Peterson *et al.* [32] are variants of qBfs for which the corresponding decision problem is NEXP-complete. In this section, we define the related coNEXP-complete complementary problem. For the definitions related to dependency quantifier Boolean formulae, we follow Virtema [41].

QBfs extend propositional logic by allowing a prenex quantification of proposition symbols. Formally, the set of *qBfs* is built from formulae of propositional logic by the following grammar:

$$\varphi ::= \exists p\, \varphi \mid \forall p\, \varphi \mid \theta,$$

where p is a propositional variable (i.e. a proposition symbol) and θ is a formula of propositional logic. The semantics for qBfs is defined via assignments $s\colon \mathrm{PROP} \to \{0, 1\}$ in the obvious way. When C is a set of propositional variables, we denote by $\vec{c}$ the canonically ordered tuple of the variables in the set C. When p is a propositional variable and $b \in \{0, 1\}$ is a truth value, we denote by $s(p \mapsto b)$ the modified assignment defined as follows:

$$s(p \mapsto b)(q) := \begin{cases} b & \text{if } q = p, \\ s(q) & \text{otherwise.} \end{cases}$$

A formula that does not have any free variables is called *closed*. We denote by QBF the set of exactly all closed qBfs.

PROPOSITION 5.6 ([38]).
The validity problem of QBF is PSPACE-complete w.r.t. $\leq_m^{\log}$.

A *simple qBf* is a closed qBf of the type $\varphi := \forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$, where θ is a propositional formula and the propositional variables p_i, q_j are all distinct. Any tuple $(C_1, \ldots, C_k)$ such that $C_1, \ldots, C_k \subseteq \{p_1, \ldots, p_n\}$ is called a *constraint* for φ. We identify such a constraint C_i via a vector $\vec{c}_i \in \{0,1\}^n$ in the obvious way: the jth position of $\vec{c}_i$ is 1 iff $p_j \in c_i$. Intuitively, a constraint $C_j = \{p_1, p_3\}$ can be seen as a dependence atom $\mathrm{dep}(p_1, p_3, q_j)$[3]. A constraint $C_j = \{p_1, p_3\}$ can be also interpreted to indicate that the semantics of $\exists g_j$ is defined, if skolemised, via a Skolem function $f_j(p_1, p_3)$.

DEFINITION 5.7
A simple qBf $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is *valid under a constraint* $(C_1, \ldots, C_k)$, if there exist functions $f_1, \ldots, f_k$ with $f_i \colon \{0,1\}^{|C_i|} \to \{0,1\}$ such that for each assignment $s \colon \{p_1, \ldots, p_n\} \to \{0,1\}$, $s(q_1 \mapsto f_1(s(\vec{c}_1)), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \models \theta$.

A *dependency quantifier Boolean formula* is a pair $(\varphi, \vec{C})$, where φ is a simple qBf and $\vec{C}$ is a constraint for φ. We say that $(\varphi, \vec{C})$ is *valid*, if φ is valid under the constraint $\vec{C}$. Let DQBF denote the set of all dependency quantifier Boolean formulae.

PROPOSITION 5.8 ([32, 5.2.2)).
The validity problem of DQBF is NEXP-complete w.r.t. $\leq_m^{\log}$.

DEFINITION 5.9
Given a simple qBf $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$, we say it is *non-valid under a constraint* $(C_1, \ldots, C_k)$, if for all functions $f_1, \ldots, f_k$ with $f_i \colon \{0,1\}^{|C_i|} \to \{0,1\}$, there exists an assignment $s \colon \{p_1, \ldots, p_n\} \to \{0,1\}$ such that $s(q_1 \mapsto f_1(s(\vec{c}_1)), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta$.

It is straightforward to see that non-validity problem of DQBF is the complement problem of the validity problem of DQBF. Accordingly, the following corollary follows.

COROLLARY 5.10
The non-validity problem of DQBF is coNEXP-complete w.r.t. $\leq_m^{\log}$.

5.3 Complexity of the validity problem is coNEXP-hard

In this section we give a reduction from the non-validity problem of DQBF to the validity problem of Minc.

THEOREM 5.11
VAL(Minc) under lax semantics is coNEXP-hard w.r.t. $\leq_m^{\log}$.

PROOF. We provide a $\leq_m^{\log}$-reduction from the non-validity problem of DQBF to the validity problem of Minc.

Recall Definition 5.9 and consider the simple qBf $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ with constraint $(C_1, \ldots, C_k)$. In our reduction we will encode all the possible modified assignments, as required by Definition 5.9, by points in Kripke models. First we enforce binary (assignment) trees of depth n in our structures. The leafs of a binary tree will correspond to the set of assignments $s \colon \{p_1, \ldots, p_n\} \to \{0,1\}$. The binary trees are forced in the standard way by modal formulae: the formula $\mathsf{branch}_{p_i} := \Diamond p_i \wedge \Diamond \neg p_i$ forces that there are ≥ 2 successor states that disagree on a

[3] See Section 6 for a definition.

proposition p_i. The formula $\mathsf{store}_{p_i} := (p_i \to \Box p_i) \wedge (\neg p_i \to \Box \neg p_i)$ is used to propagate chosen values for p_i to successors in the tree. Now define

$$\mathsf{tree}_{\bar{p},n} := \mathsf{branch}_{p_1} \wedge \bigwedge_{i=1}^{n-1} \Box^i \Big(\mathsf{branch}_{p_{i+1}} \wedge \bigwedge_{j=1}^{i} \mathsf{store}_{p_j} \Big),$$

where $\Box^i \varphi := \overbrace{\Box \cdots \Box}^{i \text{ many}} \varphi$ is the i-times concatenation of $\Box$. The formula $\mathsf{tree}_{\bar{p},n}$ forces a complete binary assignment tree of depth n for proposition symbols $p_1, \ldots, p_n$. More precisely, each of the nodes in the tree of depth n is labelled by some subset of the propositions $\{p_1, \ldots, p_n\}$, and each subset of $\{p_1, \ldots, p_n\}$ is the label of some such node. Notice that $\mathsf{tree}_{\bar{p},n}$ is an ML-formula and consequently flat (see Proposition 4.3). Let $\ell := \max\{|C_1|, \ldots, |C_k|\}$. Then define

$$\begin{aligned} \varphi_{\text{struc}} :=& \mathsf{tree}_{\bar{p},n} \wedge \Box^n \big(\mathsf{tree}_{\bar{t},\ell} \big) \wedge \Box^{n+\ell} \big((p_\theta \leftrightarrow \theta) \wedge p_\top \wedge \neg p_\bot \big) \\ & \wedge \Box^n \Big(\bigwedge_{1 \le i \le \ell} \Box^i \big(\bigwedge_{1 \le j \le n} \mathsf{store}_{p_j} \wedge \bigwedge_{1 \le r \le k} \mathsf{store}_{q_r} \big) \Big). \end{aligned}$$

The formula φ_{struc} enforces the full binary assignment tree w.r.t. the p_is, enforces in its leaves trees of depth ℓ for variables t_i, identifies the truth of θ by a proposition p_θ at the depth $n + \ell$ as well as 1 by $p_\top$ and 0 by $p_\bot$ and then stores the values of the p_js and q_rs consistently in their subtrees of relevant depth. The points at depth n are used to encode the modified assignments of Definition 5.9.

Now consider some particular Kripke model with the structural properties described above. Shift your attention to those points in the enforced tree that are in depth n. Note first, all assignments for proposition symbols $p_1, \ldots, p_n$ are present. In fact, the number of points corresponding to some particular assignment can by anything ≥ 1. Values for the proposition symbols q_j and consequently for the functions f_j arise from the particular model; essentially, since we are considering validity, all possible values will be considered. In fact, in some particular models, the values of q_js are not functionally determined according to the related constraint C_j. We will next define a formula that will deal with those models in which, for some j, the values for q_j do not give rise to a function f_j as in Definition 5.9. These unwanted models have to be 'filtered' out by the formula through satisfaction. This violation is expressed via φ_{cons} defined as follows. Below we let $n_j = |C_j|$.

$$\varphi_{\text{cons}} := \bigvee_{\substack{1 \le j \le k, \\ C_j = \{p_{i_1}, \ldots, p_{i_{n_j}}\}}} \big((t_1 \cdots t_{n_j} p_\bot \subseteq p_{i_1} \cdots p_{i_{n_j}} q_j) \wedge (t_1 \cdots t_{n_j} p_\top \subseteq p_{i_1} \cdots p_{i_{n_j}} q_j) \big). \tag{3}$$

Assume that $p_\top$ and $p_\bot$ correspond to the constant values 1 and 0, respectively. Moreover, for the time being, suppose that the values for the proposition symbols t_i have been existentially quantified (we will later show how this is technically done). Now the formula φ_{cons} essentially states that there exists a q_j that does not respect the constraint C_j.

Finally define

$$\varphi_{\text{non-val}} := \varphi^{\perp}_{\text{struc}} \vee \Big(\varphi_{\text{struc}} \wedge \Box^n \big(\Diamond^\ell (\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta) \big) \Big). \tag{4}$$

By $\varphi^{\perp}_{\text{struc}}$, we denote the negation normal form of the ML-formula $\neg \varphi_{\text{struc}}$. An important observation is that since φ_{struc} is an ML-formula, it is flat. Now the formula $\varphi_{\text{non-val}}$ is valid if

and only if

$$\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta)\big) \tag{5}$$

for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}. Let us now return to the formula (3). There, we assumed that the proposition symbols t_i had been quantified and that the symbols $p_\top$ and $p_\bot$ correspond to the logical constants. The latter part we already dealt with in the formula φ_{struc}. Recall that φ_{struc} forces full binary assignment trees for the t_is that start from depth n. The quantification of the t_is is done by selecting the corresponding successors by the diamonds $\Diamond^\ell$ in the formula (4). If $\mathfrak{M}, T$ is such that, for some j, q_j does not respect the constraint C_j, we use $\Diamond^\ell$ to guess a witness for the violation. It is then easy to check that the whole team obtained by evaluating the diamond prefix satisfies the formula φ_{cons}. On the other hand, if $\mathfrak{M}, T$ is such that for each j the value of q_j respects the constraint C_j, then the subformula $p_\bot \subseteq p_\theta$ forces that there exists a point w in the team obtained from T by evaluating the modalities in (5) such that $\mathfrak{M}, \{w\} \not\models p_\theta$. In our reduction this means that w gives rise to a propositional assignment that falsifies θ as required in Definition 5.9.

It is now quite straightforward to show that a simple qBf $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is *non-valid under a constraint* $(C_1, \ldots, C_k)$ iff the Minc-formula $\varphi_{\text{non-val}}$ obtained as described above is valid.

In the following we show the correctness of the constructed reduction. By the observation made in (5) it suffices to show the following claim:

CLAIM $\forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is non-valid under $(C_1, \ldots, C_k)$ iff $\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta)\big)$ for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}.

□

PROOF OF CLAIM '⇒': assume that the formula $\varphi := \forall p_1 \cdots \forall p_n \exists q_1 \cdots \exists q_k \theta$ is non-valid under the constraint $(C_1, \ldots, C_k)$. As a consequence, for every sequence of functions $f_1, \ldots, f_k$ of appropriate arities there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that

$$s(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta. \tag{6}$$

We will show that

$$\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\bot \subseteq p_\theta)\big), \tag{7}$$

for each team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}.

Let $\mathfrak{M}, T$ be an arbitrary team pointed Kripke model that satisfies the required structural properties. Denote by S the team obtained from T after evaluating the first n $\Box$-symbols in (7). Note that each tuple of values assigned to $\vec{p} := (p_1, \ldots, p_n)$ is realized in S as the tree structure enforces all possible assignments over $\vec{p}$. Due to the forced structural properties, S and any of the teams obtainable from S by evaluating the k $\Diamond$-symbols in (7) realize exactly the same assignments for $\{p_1, \ldots, p_n, q_1, \ldots q_k\}$. Let S_k denote the set of exactly all points reachable from S by paths of length exactly k. For each point w denote by $w(\vec{q})$ the value of $\vec{q}$ in the world w. Note that for every ℓ-tuple of bits $\vec{b}$ and every point $w \in S$ there exists a point $v \in S_k$ such that $v(p_1, \ldots, p_n, q_1, \ldots q_k) = w(p_1, \ldots, p_n, q_1, \ldots q_k)$ and $v(t_1, \ldots, t_\ell) = \vec{b}$. Moreover, for any fixed $\vec{b}$, the team

$$\{w \in S_k \mid w(t_1, \ldots, t_\ell) = \vec{b}\}$$

is obtainable from S by evaluating the k $\Diamond$-symbols in (7). We have two cases:

1. There exists a constraint C_i, $1 \leq i \leq k$, and points $w, w' \in S$ with $w(\vec{c}_i) = w'(\vec{c}_i)$ but $w(q_i) \neq w'(q_i)$. Now let S' be a team obtained from S by evaluating the k $\Diamond$-symbols in (7) such that, for every $w' \in S'$, $w(t_1, \ldots, t_\ell)$ is an expansion of $w(\vec{c}_i)$. Now clearly $\mathfrak{M}, S' \models \varphi_{\text{cons}}$ and as a consequence $\mathfrak{M}, S \models \Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta)$. From this (7) follows.
2. For each C_i, $1 \leq i \leq k$ and every $w, w' \in S$, if $w(\vec{c}_i) = w'(\vec{c}_i)$ then $w(q_i) = w'(q_i)$. Let $f_1, \ldots, f_k$ be some functions that arise from the fact that the constraints $(C_1, \ldots, C_k)$ are satisfied in S. Since, by assumption, φ is non-valid under the constraint $(C_1, \ldots, C_k)$, it follows that there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that (6) is true. Now recall that each tuple of values assigned to $\vec{p} := (p_1, \ldots, p_n)$ is realized in S. Accordingly, in particular, s and $s(q_1 \mapsto f_1(s(\vec{c}_1)), \ldots, q_k \mapsto f_k(s(\vec{c}_k)))$ are realized in S. For this reason $\mathfrak{M}, S \models \Diamond^\ell(p_\perp \subseteq p_\theta)$, from which (7) follows in a straightforward manner.

'$\Leftarrow$': Assume that $\mathfrak{M}, T \models \Box^n\big(\Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta)\big)$ for every team pointed Kripke model $\mathfrak{M}, T$ that satisfies the structural properties forced by φ_{struc}. We need to show that φ is non-valid under the constraint $(C_1, \ldots, C_k)$. In order to show this, let $f_1, \ldots, f_k$ be arbitrary functions with arities that correspond to the constraint $(C_1, \ldots, C_k)$. Let $\mathfrak{M}, T$ be a team pointed Kripke model and S a team of $\mathfrak{M}$ such that

a) $\mathfrak{M}, T$ satisfies the structural properties forced by φ_{struc},
b) S is obtained from T by evaluating the n $\Box$-symbols,
c) $f_i\big(w(\vec{c}_i)\big) = w(q_i)$, for each $w \in S$ and $1 \leq i \leq k$.

It is easy to check that such a model and teams always exist. From the assumption we then obtain that

$$\mathfrak{M}, S \models \Diamond^\ell(\varphi_{\text{cons}} \vee p_\perp \subseteq p_\theta). \tag{8}$$

But since the values of q_is, by construction, do not violate the constraint $(C_1, \ldots, C_k)$, we obtain, with the help of the structural properties, that for (8) to hold is must be the case that $\mathfrak{M}, S_k \models p_\perp \subseteq p_\theta$, where S_k is some team obtained from S by evaluating the k $\Diamond$-symbols in (8). But this means that there exists an assignment $s\colon \{p_1, \ldots, p_n\} \to \{0, 1\}$ such that

$$s(q_1 \mapsto f_1\big(s(\vec{c}_1)\big), \ldots, q_k \mapsto f_k(s(\vec{c}_k))) \not\models \theta. \tag{9}$$

Consequently, the claim is true. □

In order to compute $\varphi_{\text{non-val}}$ two binary counters bounded above by $n+k+\ell$ need to be maintained. Note that $\log(n + k + \ell)$ is logarithmic with respect to the input length. That being the case, the reduction is computable in logspace and the lemma applies.

The construction in the previous proof works also for strict semantics. In the proof of the claim a small adjustment is needed to facilitate the strict semantics of diamond. As a result we obtain the following.

COROLLARY 5.12
VAL(Minc) under strict semantics is coNEXP-hard w.r.t. $\leq_m^{\log}$.

PROOF OF SKETCH We discuss the proof of Theorem 5.11 regarding the effect of shifting from lax to strict semantics with respect to $\Diamond$ and $\vee$:

- For ML-subformulae strict and lax semantics coincide (Proposition 4.3).
- φ_{cons}: no change is needed.

- $\varphi_{\text{non-val}}$: the outermost disjunction is always a partition enforced by the formula φ_{struc} and its negation, and hence this disjunction behaves exactly the same under strict semantics and lax semantics.
- Regarding equation (5), in the case some constraint is violated, strict diamonds will find a witness for the violation (under lax semantics, multiple violations can be found but one suffices).
- In the proof of the claim on page 20, in general, the set $S^* := \{w \in S_k \mid w(t_1, \ldots, t_\ell) = \vec{b}\}$ is not obtainable from S using strict diamonds. However, a subset $H \subseteq S^*$ of size $|S|$ such that $\{w(\vec{p}, \vec{q}, \vec{t}) \mid w \in S^*\} = \{w(\vec{p}, \vec{q}, \vec{t}) \mid w \in H\}$ is reachable, and that suffices. □

While the exact complexities of the problems VAL(Minc) and VAL(EMinc) remain open, it is straightforward to establish that the two complexities coincide. In the proof of the theorem below, we introduce fresh proposition symbols for each modal formula that appears as a parameter for an inclusion atom. We then replace these formulae by the fresh proposition symbols and separately force, by using an ML-formula, that the extensions of the proposition symbols and modal formulae are the same, respectively.

THEOREM 5.13
Let C be a complexity class that is closed under polynomial time reductions. Then VAL(Minc) under lax (strict) semantics in complete for C if and only if VAL(EMinc) under lax (strict) semantics in complete for C.

PROOF. Let φ be a formula of EMinc and k the modal depth of φ. Let $\varphi_1, \ldots, \varphi_n$ be exactly those subformulae of φ that occur as a parameter of some inclusion atom in φ and let $p_1, \ldots, p_n$ be distinct fresh proposition symbols. Define

$$\varphi_{\text{subst}} := \left(\bigwedge_{0 \le i \le k} \Box^i \bigwedge_{1 \le j \le n} (p_j \leftrightarrow \varphi_j) \right),$$

$$\varphi^* := \varphi^{\perp}_{\text{subst}} \vee (\varphi_{\text{subst}} \wedge \varphi^+),$$

where $\varphi^{\perp}_{\text{subst}}$ denotes the negation normal form of $\neg\varphi_{\text{subst}}$ and φ^+ is the formula obtained from φ by simultaneously substituting each φ_i by p_i. It is easy to check that φ is valid if and only if the Minc formula φ^* is. Clearly φ^* is computable from φ in polynomial time. □

6 Conclusion

In this paper, we investigated the computational complexity of model checking and validity for propositional and modal inclusion logic with the aim to eventually complete the complexity landscape of the basic decision problems of the most prominent propositional and modal logics in the team semantics setting. Tables 4–6 give an overview of the current state of research for the most important decision problems in both strict and lax semantics setting. Furthermore, we identify the left open cases for further research. In the tables, ATIME-*ALT*(*exp*, *poly*) refers to alternating exponential time with polynomially many alternations, and TOWER(*poly*) refers to the class of problems that can be decided by a deterministic Turing machine with runtimes that can be described by an exponential tower of 2s of polynomial height.

As the tables also mention atoms that have not been considered elsewhere in this paper, we introduce them shortly:

TABLE 4 Complexity of model checking. All results are completeness results. Whenever strict and lax semantics coincide, only a single complexity result is shown which is surrounded by dashes.

	PL model checking		ML model checking	
Operator	Strict	Lax	Strict	Lax
ø	—— NC^1 [2] ——		—— P [4, 36] ——	
dep(·)	—— NP [8] ——		—— NP [8] ——	
⊆	NP [Thm. 17]	P [Thm. 13]	NP [Thm. 29]	P [Thm. 28]
⊥	NP⋆	NP [16]	NP⋆	NP [24]
∼	PSPACE⋆	PSPACE [16, 31]	PSPACE⋆	PSPACE [31]

⋆: Proof for lax semantics works also for strict semantics.

TABLE 5 Complexity of satisfiability. Complexity classes refer to completeness results; '-h' denotes hardness. Whenever strict and lax semantics coincide, only a single complexity result is shown, which is surrounded by dashes.

	PL satisfiability problem		ML Satisfiability Problem	
Operator	Strict	Lax	strict	lax
ø	—— NP [5, 27] ——		—— PSPACE [26] ——	
dep(·)	—— NP [28] ——		—— NEXP [37] ——	
⊆	EXP [19]	EXP [18]	EXP [19]	EXP [18]
⊥	NP⋆	NP [16]	NEXP⋆	NEXP [24]
∼	ATIME-*ALT* $(exp, poly)$⋆	ATIME-*ALT* $(exp, poly)$ [16]	TOWER(*poly*)-h⋆	TOWER(*poly*) [30]

⋆: Proof for lax semantics works also for strict semantics.

TABLE 6 Complexity of validity. Complexity classes refer to completeness results, '-h' denotes hardness and '∈' denotes containment. Whenever strict and lax semantics coincide, only a single complexity result is shown, which is surrounded by dashes.

	PL validity problem		ML validity problem	
Operator	Strict	Lax	Strict	Lax
ø	—— coNP [5, 27] ——		—— PSPACE [26] ——	
dep(·)	—— NEXP [41] ——		—— NEXP [13] ——	
⊆	coNP [Thm. 3.1]	coNP [16]	coNEXP-h [Cor. 5.12]	coNEXP-h [Thm. 5.11]
⊥	$\in \mathsf{coNEXP}^{\mathsf{NP}\star}$	$\in \mathsf{coNEXP}^{\mathsf{NP}}$ [16]	?	?
∼	ATIME-*ALT* $(exp, poly)$⋆	ATIME-*ALT* $(exp, poly)$ [16]	TOWER(*poly*)-h⋆	TOWER(*poly*) [30]

⋆: Proof for lax semantics works also for strict semantics. ?: No nontrivial result is known.

Let $\vec{p}$, $\vec{q}$ and $\vec{r}$ be tuples of proposition symbols and q a proposition symbol. Then $\mathrm{dep}(\vec{p}, q)$ is a *propositional dependence atom* and $\vec{q} \perp_{\vec{p}} \vec{r}$ is a *conditional independence atom* with the following semantics:

$$X \models \mathrm{dep}(\vec{p}, q) \;\Leftrightarrow\; \forall s, t \in X : s(\vec{p}) = t(\vec{p}) \text{ implies } s(q) = t(q).$$

$$X \models \vec{q} \perp_{\vec{p}} \vec{r} \;\Leftrightarrow\; \forall s, t \in X : \text{ if } s(\vec{p}) = t(\vec{p}), \text{then } \exists u \in X : u(\vec{p}\vec{q}) = s(\vec{p}\vec{q}) \text{ and } u(\vec{r}) = t(\vec{r}).$$

Intuitively, $\vec{q} \perp_{\vec{p}} \vec{r}$ states that $\vec{q}$ and $\vec{r}$ are informationally independent for any fixed value for $\vec{p}$. We also consider the contradictory negation $\sim$ in our setting. The semantics of $\sim$ is defined as follows:

$$X \models \sim\varphi \text{ if and only if } X \not\models \varphi.$$

The semantics for these atoms and for the contradictory negation in the modal setting is defined analogously. When $\mathcal{C}$ is a set of atoms, we denote by $\mathsf{PL}(\mathcal{C})$ and $\mathsf{ML}(\mathcal{C})$ the extensions of PL and ML, in the team semantics setting, by the atoms in $\mathcal{C}$, respectively.

As depicted in Tables 4 and 5, the model checking and satisfiability problems for our logics have been completely characterized. The last open case, the complexity of satisfiability of $\mathsf{ML}(\sim)$, was recently solved by Lück [30]; there, only lax semantics is considered, but the proof of the hardness result works directly also for strict semantics. Concerning the validity problem, the major open problem is the complexity of logics with $\perp$; here, only a partial result in the propositional setting is known. We conclude with a list of open problems:

- It is known that the complexity of validity of $\mathsf{PL}(\perp)$ is between NEXP and $\mathsf{coNEXP}^{\mathsf{NP}}$. What is the precise complexity?
- What is the complexity of validity of $\mathsf{ML}(\perp)$?
- We showed that the validity problem for Minc and EMinc are coNEXP-hard for both lax and strict semantics. Can we show a corresponding upper bound?
- Tables 4–6 indicate that complexities of lax and strict semantics coincide above the complexity class P. Can we discover a meta theorem from this observation?

References

[1] P. Blackburn, M. de Rijke and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[2] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th STOC*, pp. 123–131, 1987.

[3] M. A. Casanova, R. Fagin and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *Journal of Computer and Systems Sciences*, **28**, 29–59, 1984. doi:10.1016/0022-0000(84)90075-8.

[4] E. Clarke, E. A. Emerson and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM ToPLS*, **8**, 244–263, 1986.

[5] S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd STOC*, pp. 151–158, 1971.

[6] A. Durand, M. Hannula, J. Kontinen, A. Meier and J. Virtema. Approximation and dependence via multiteam semantics. *Annals of Mathematics and Artificial Intelligence*, **83**, 297–320, 2018. doi:10.1007/s10472-017-9568-4.

[7] A. Durand, J. Kontinen and H. Vollmer. Expressivity and complexity of dependence logic. In *Dependence Logic: Theory and Applications*, S. Abramsky, J. Kontinen, J. Väänänen, and H. Vollmer, eds, pp. Birkhäuser, Basel, 5–32, 2016.

[8] J. Ebbing and P. Lohmann. Complexity of model checking for modal dependence logic. In *38th Proc. SOFSEM*, pp. 226–237, 2012.

[9] P. Galliani. Inclusion and exclusion dependencies in team semantics—on some logics of imperfect information. *Annals of Pure and Applied Logic*, **163**, 68–84, 2012.
[10] P. Galliani and L. Hella. Inclusion logic and fixed point logic. In *Proc. 22nd CSL*, LIPIcs, pp. 281–295, 2013.
[11] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
[12] L. M. Goldschlager. The monotone and planar circuit value problems are log-space complete for P. *SIGACT News*, **9**, 25–29, 1977.
[13] M. Hannula. Validity and Entailment in Modal and Propositional Dependence Logics. In Valentin Goranko and Mads Dam, eds, *26th EACSL Annual Conference on Computer Science Logic (CSL*2017*)*, vol. 82 *of Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 28:1–28:17, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7669, doi:10.4230/LIPIcs.CSL.2017.28.
[14] M. Hannula and J. Kontinen. A finite axiomatization of conditional independence and inclusion dependencies. *Information and Computation*, **249**, 121–137, 2016. doi:10.1016/j.ic.2016.04.001.
[15] M. Hannula and J. Kontinen., and J. Virtema. Polyteam semantics. In *Logical Foundations of Computer Science—International Symposium, LFCS*2018, *Deerfield Beach, FL, USA, January 8–11, 2018, Proceedings*, pp. 190–210, 2018. doi: 10.1007/978-3-319-72056-2_12.
[16] M. Hannula, J. Kontinen, J. Virtema and H. Vollmer. Complexity of propositional logics in team semantic. *ACM Transactions on Computational Logic*, **19**, 2:1–2:14, 2018. doi: 10.1145/3157054.
[17] L. Hella, A. Kuusisto, A. Meier and J. Virtema. Model Checking and Validity in Propositional and Modal Inclusion Logics. In Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin, eds, *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, vol. 83 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 32:1–32:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2017/8100, doi:10.4230/LIPIcs.MFCS.2017.32.
[18] L. Hella, A. Kuusisto, A. Meier and H. Vollmer. Modal inclusion logic: being lax is simpler than being strict. In *Proc. 40th MFCS*, pp. 281–292, 2015.
[19] L. Hella, A. Kuusisto, A. Meier and H. Vollmer. Satisfiability of modal inclusion logic: lax and strict semantics. *CoRR*, 2015. arXiv: 1504.06409.
[20] L. Hella and J. Stumpf. The expressive power of modal logic with inclusion atoms. In *Proc. 6th GandALF*, pp. 129–143, 2015.
[21] W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, **5**, 539–563, 1997.
[22] N. Immerman. Relational queries computable in polynomial time (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, 147–152, New York, NY, USA, 1982. *ACM*. doi: 10.1145/800070.802187.
[23] H. Köhler and S. Link. Inclusion dependencies and their interaction with functional dependencies in SQL. *Journal of Computer and Systems Sciences*, **85**, 104–131, 2017. doi: 10.1016/j.jcss.2016.11.004.
[24] J. Kontinen, J.-S. Müller, H. Schnoor and H. Vollmer. Modal independence logic. *Journal of Logic and Computation*, **27**, 1333–1352, 2017. URL, https://doi.org/10.1093/logcom/exw019.
[25] A. Krebs, A. Meier, J. Virtema and M. Zimmermann. Team Semantics for the Specification and Verification of Hyperproperties. In Igor Potapov, Paul Spirakis, and James Worrell, eds, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS*

2018), vol. 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 10:1–10:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9592, doi:10.4230/LIPIcs.MFCS.2018.10.

[26] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, **6**, 467–480, 1977.

[27] L. A. Levin. Universal sorting problems. *Problems of Information Transmission*, **9**, 265–266, 1973.

[28] P. Lohmann and H. Vollmer. Complexity results for modal dependence logic. *Studia Logica*, **101**, 343–366, 2013.

[29] M. Lück. Axiomatizations of team logics. *Annals of Pure and Applied Logic*, **169**, 928–969, 2018. doi: 10.1016/j.apal.2018.04.010.

[30] M. Lück. Canonical models and the complexity of modal team logic. In D. R. Ghica and A. Jung, eds, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4–7, 2018, Birmingham, UK*, vol. 119 of *LIPIcs*, pp. 30:1–30:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.30.

[31] J.-S. Müller. *Satisfiability and Model Checking in Team Based Logics*. PhD Thesis, Leibniz University of Hannover, 2014.

[32] G. Peterson, J. Reif and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics With Applications*, **41**, 957–992, 2001.

[33] K. Sano and J. Virtema. Axiomatizing propositional dependence logics. In Stephan Kreutzer, ed, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7–10, 2015, Berlin, Germany*, vol. 41 of *LIPIcs*, pp. 292–307. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:https://doi.org/10.4230/LIPIcs.CSL.2015.292.

[34] K. Sano and J. Virtema. Characterizing frame definability in team semantics via the universal modality. *In Proc. of WoLLIC*, **2015**, pp. 140–155, 2015.

[35] K. Sano and J. Virtema. Characterizing relative frame definability in team semantics via the universal modality. *In Proc. of WoLLIC*, **2016**, pp. 392–409, 2016.

[36] P. Schnoebelen. The complexity of temporal logic model checking. In *Proc. 4th AiML*, pp. 393–436. 2002.

[37] M. Sevenster. Model-theoretic and computational properties of modal dependence logic. *Journal of Logic and Computation*, **19**, 1157–1173, 2009.

[38] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Proc. 5th STOC*, pp. 1–9, New York, NY, USA, ACM, 1973.

[39] J. Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

[40] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82,137–146, ACM, New York, NY, USA, 1982. doi:10.1145/800070.802186.

[41] J. Virtema. Complexity of validity for propositional dependence logics. *Information and Computation*, 253224–236, 2017. doi:10.1016/j.ic.2016.07.008.

[42] H. Vollmer. *Introduction to Circuit Complexity—A Uniform Approach*. Texts in Theoretical Computer Science. Springer, Berlin Heidelberg, 1999.

[43] F. Yang and J. Väänänen. Propositional team logics. *Annals of Pure and Applied Logic*, **168**, 1406–1441, 2017. doi: 10.1016/j.apal.2017.01.007.

Received 19 March 2019